Terraform Task

Name:Shaik Khaja Basha
Batch : Batch 11
Date : 23.07.2025
Task : datatypes

---

1. **Define all datatypes using random provider**

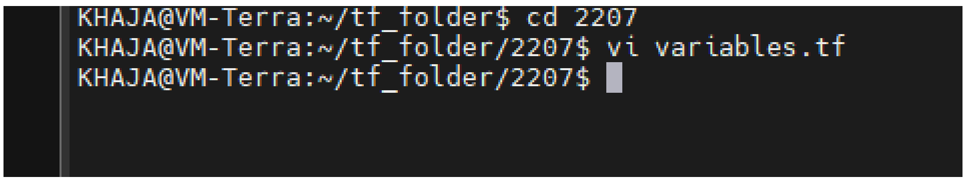
Ans :

Create a file name variable.tf using vi variable.tf and enter the details

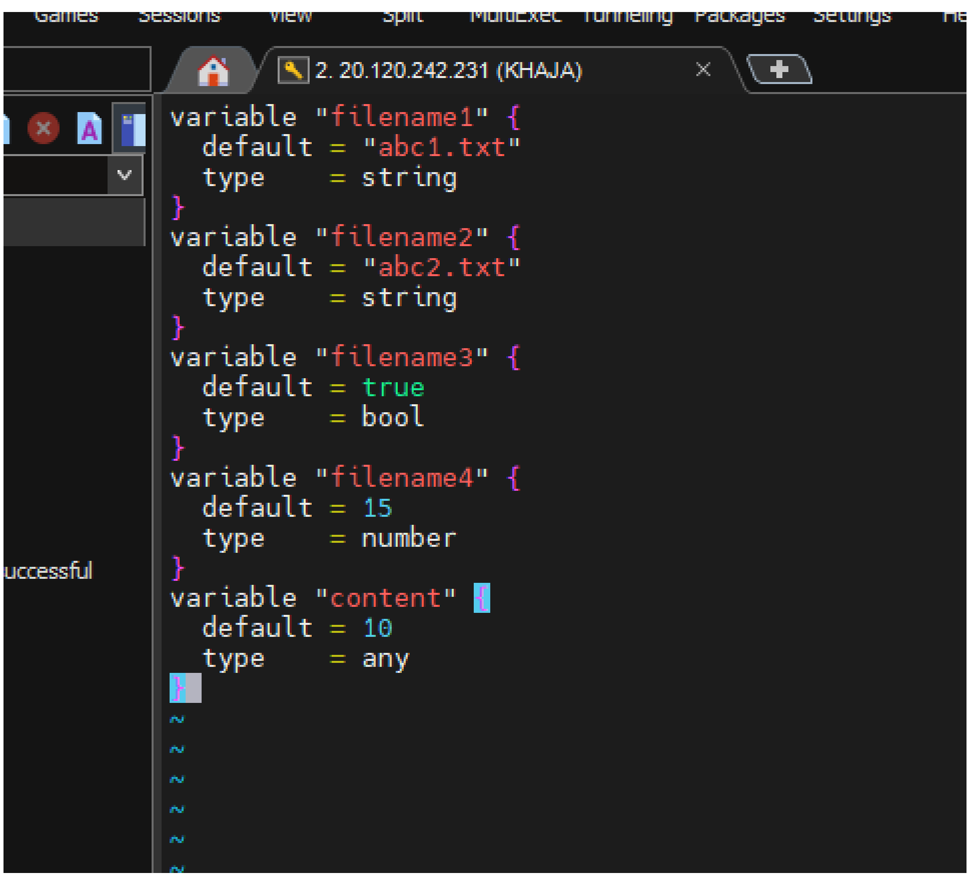
Primitive data types

```
# 1. String Variable
variable "filename1" {
  default = "abc1.txt"
  type    = string
}

# 2. String Variable (same as above, but explicitly typed)
variable "filename2" {
  default = "abc2.txt"
  type    = string
}

# 3. Bool Variable – it’s a Boolean type
variable "filename3" {
  default = true
  type    = bool
}
```

```
# 4. Number Variable
variable "filename4" {
  default = 15
  type    = number
}


# Content can be of any type
variable "content" {
  default = 10
  type    = any
}
```

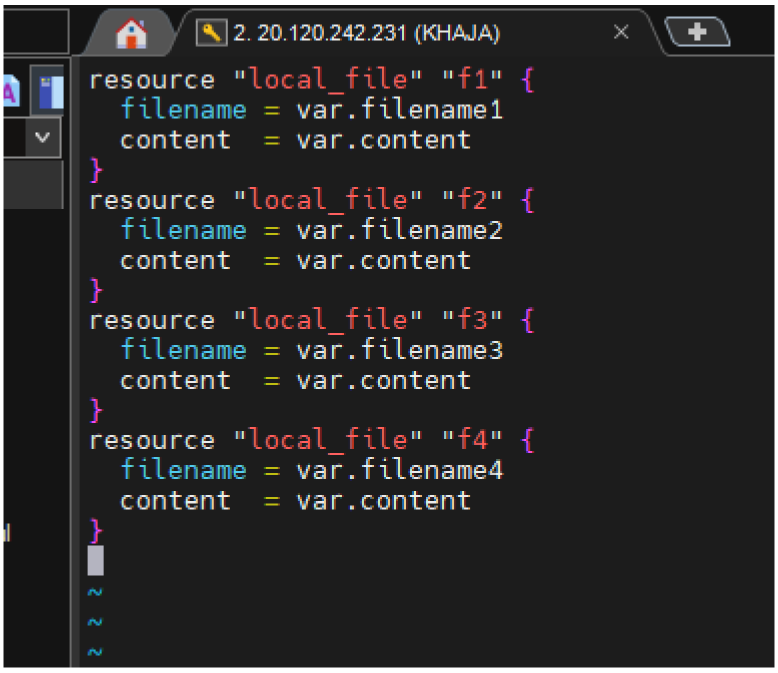


```
KHAJA@VM-Terra:~/tf_folder$ cd 2207
KHAJA@VM-Terra:~/tf_folder/2207$ vi variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$
```



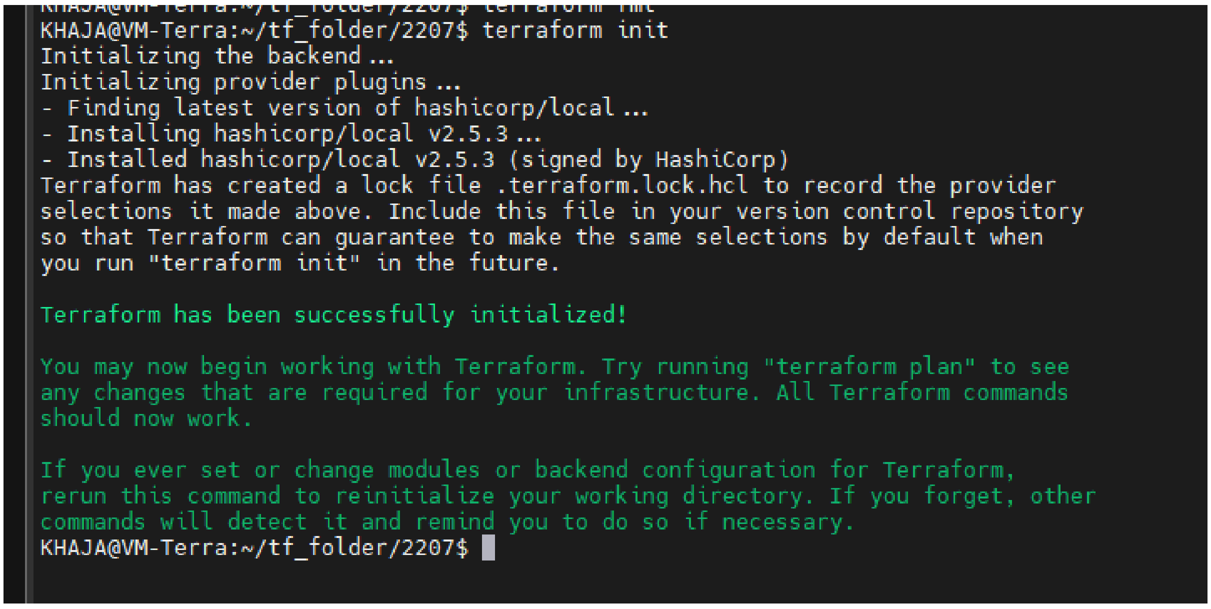
```
variable "filename1" {
  default = "abc1.txt"
  type    = string
}
variable "filename2" {
  default = "abc2.txt"
  type    = string
}
variable "filename3" {
  default = true
  type    = bool
}
variable "filename4" {
  default = 15
  type    = number
}
variable "content" {
  default = 10
  type    = any
}
~
~
~
~
~
~
```

Resources are



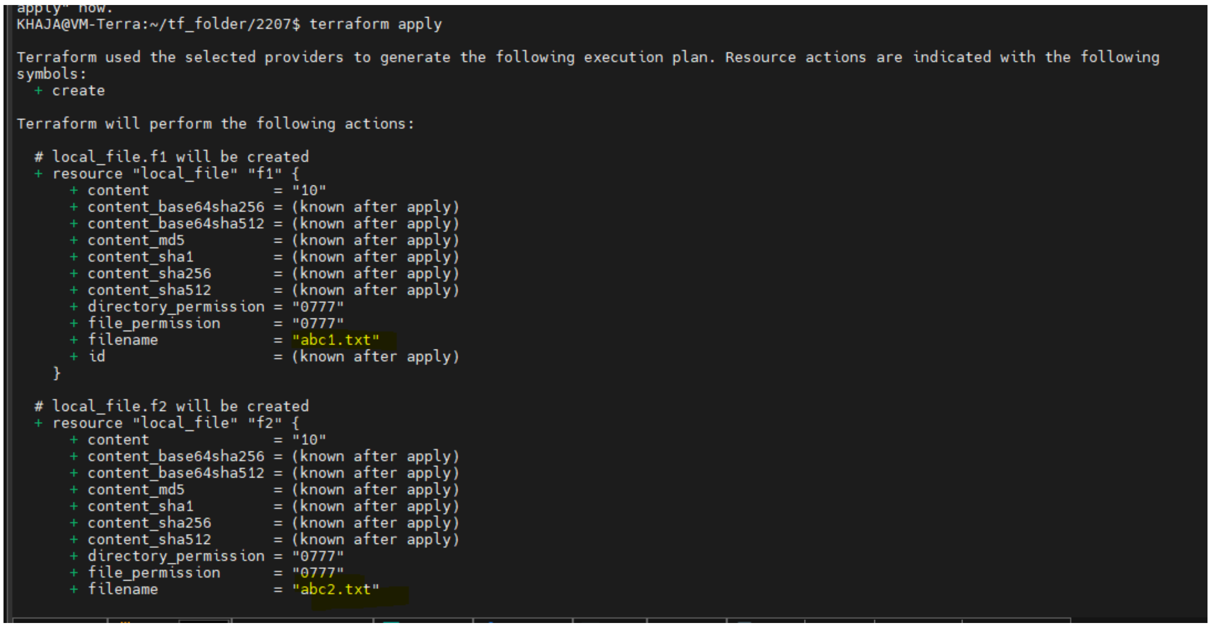
```
resource "local_file" "f1" {
  filename = var.filename1
  content  = var.content
}
resource "local_file" "f2" {
  filename = var.filename2
  content  = var.content
}
resource "local_file" "f3" {
  filename = var.filename3
  content  = var.content
}
resource "local_file" "f4" {
  filename = var.filename4
  content  = var.content
}
~
~
~
```

Execute the command terraform init

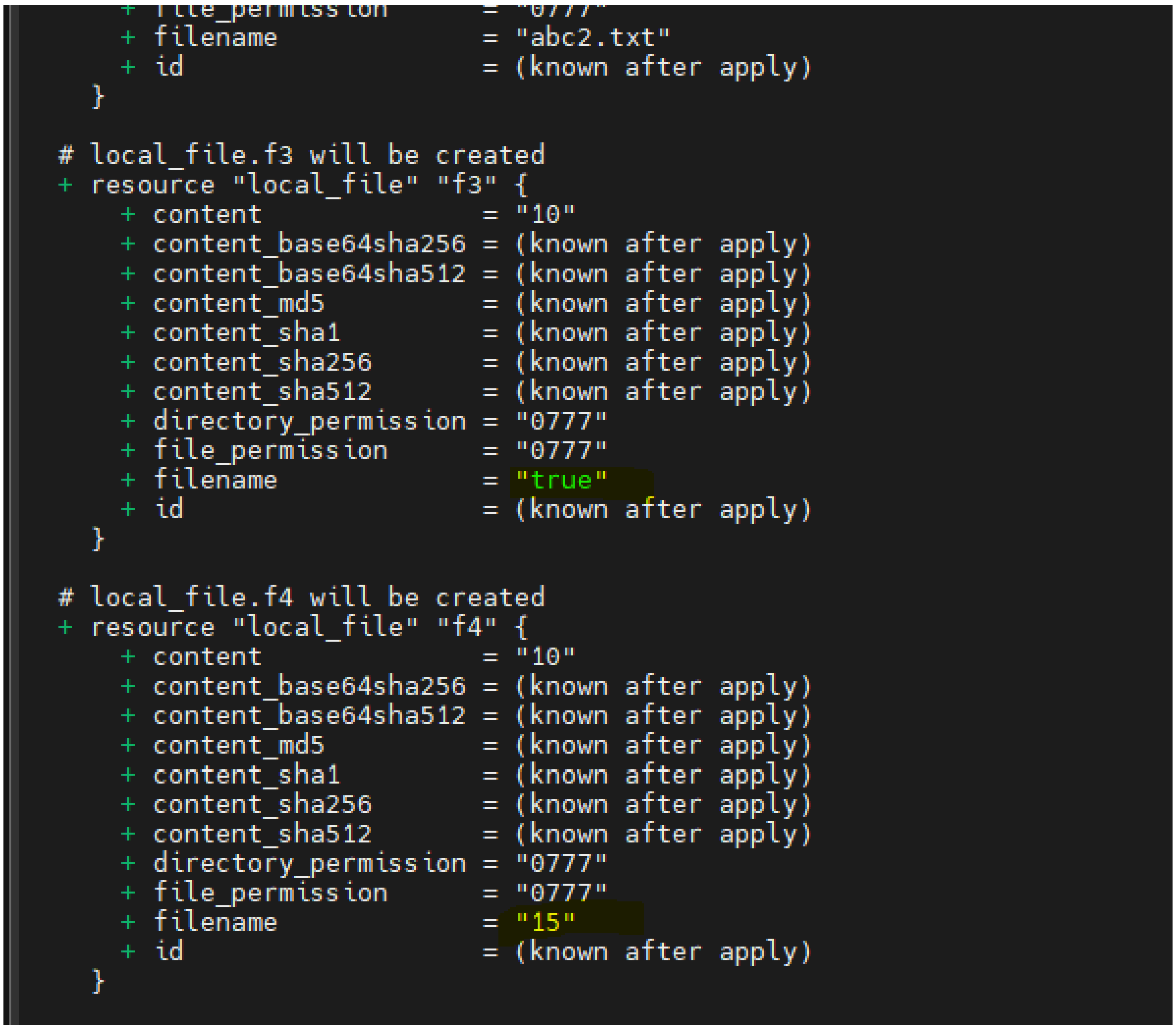


```
KHAJA@VM-Terra:~/tf_folder/2207$ terraform init
Initializing the backend ...
Initializing provider plugins ...
- Finding latest version of hashicorp/local ...
- Installing hashicorp/local v2.5.3 ...
- Installed hashicorp/local v2.5.3 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/2207$
```

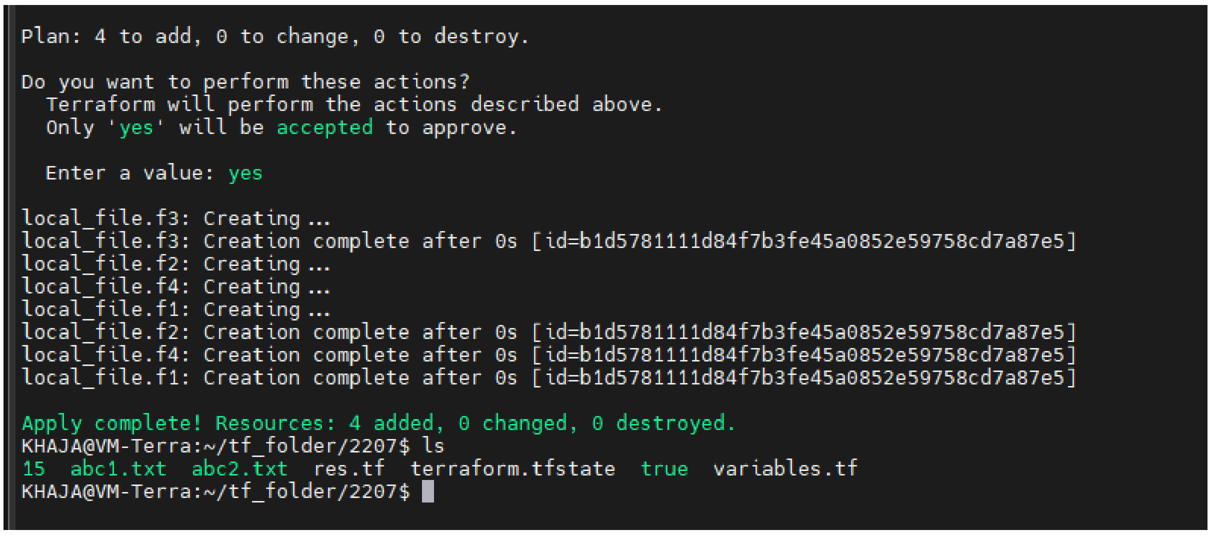
Execute terraform apply



```
apply" now.
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.f1 will be created
  + resource "local_file" "f1" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc1.txt"
      + id                   = (known after apply)
    }

  # local_file.f2 will be created
  + resource "local_file" "f2" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "abc2.txt"
```



```
      + file_permission      = "0777"
      + filename             = "abc2.txt"
      + id                   = (known after apply)
    }

  # local_file.f3 will be created
  + resource "local_file" "f3" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "true"
      + id                   = (known after apply)
    }

  # local_file.f4 will be created
  + resource "local_file" "f4" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "15"
      + id                   = (known after apply)
    }
```



```
Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.f3: Creating...
local_file.f3: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f2: Creating...
local_file.f4: Creating...
local_file.f1: Creating...
local_file.f2: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$ ls
15  abc1.txt  abc2.txt  res.tf  terraform.tfstate  true  variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$
```

Here we can see four resources which we have given is created

Terraform Complex / Composite / Advanced Data Types:

creating of resources :

# creating resource using set type

```
resource "local_file" "f5" {
  filename = var.filename5[0] # (here its using via index)
  content  = var.content
}
resource "local_file" "f6" {
  filename = var.filename6[2]
  content  = var.content
}
```

# creating resource using map type

```
resource "local_file" "f7" {
  filename = var.filename7.name # (here its using via key name)
  content  = var.content
}
resource "local_file" "f8" {
  filename = var.filename8.id
  content  = var.content
}
```

For Variable Declaration:

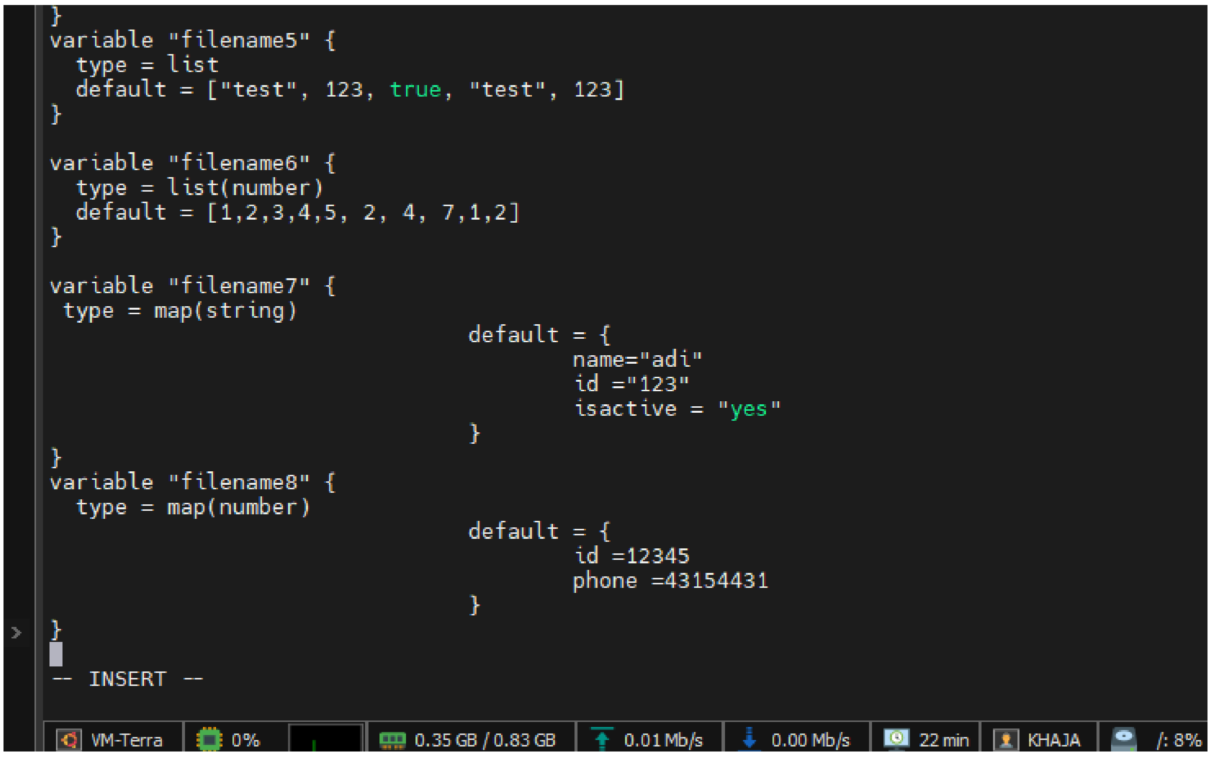
# list

```
variable "filename5" {
  type = list
  default = ["test", 123, true, "test", 123]
}

variable "filename6" {
  type = list(number)
  default = [1,2,3,4,5, 2, 4, 7,1,2]
}
```

# For Map

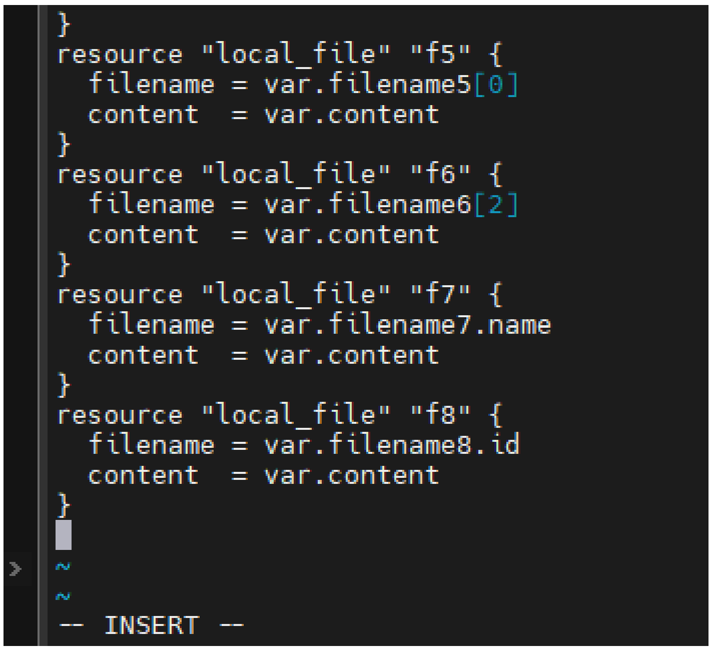
```
variable "filename7" {
 type = map(string)
          default = {
          name="adi"
          id ="123"
          isactive = "yes"
                    }
}
variable "filename8" {
  type = map(number)
                                        default = {
                                                  id =12345
                                                  phone =43154431
                                        }
}
```

Variable are mapped to variable file like vi variable.tf



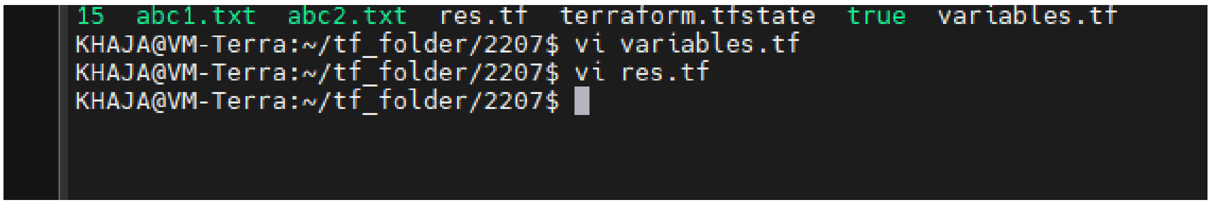
```
}
variable "filename5" {
  type = list
  default = ["test", 123, true, "test", 123]
}

variable "filename6" {
  type = list(number)
  default = [1,2,3,4,5, 2, 4, 7,1,2]
}

variable "filename7" {
 type = map(string)
                                default = {
                                        name="adi"
                                        id ="123"
                                        isactive = "yes"
                                }
}
variable "filename8" {
  type = map(number)
                                default = {
                                        id =12345
                                        phone =43154431
                                }
}

-- INSERT --
```

Adding resources to the resource file “vi res.tf”



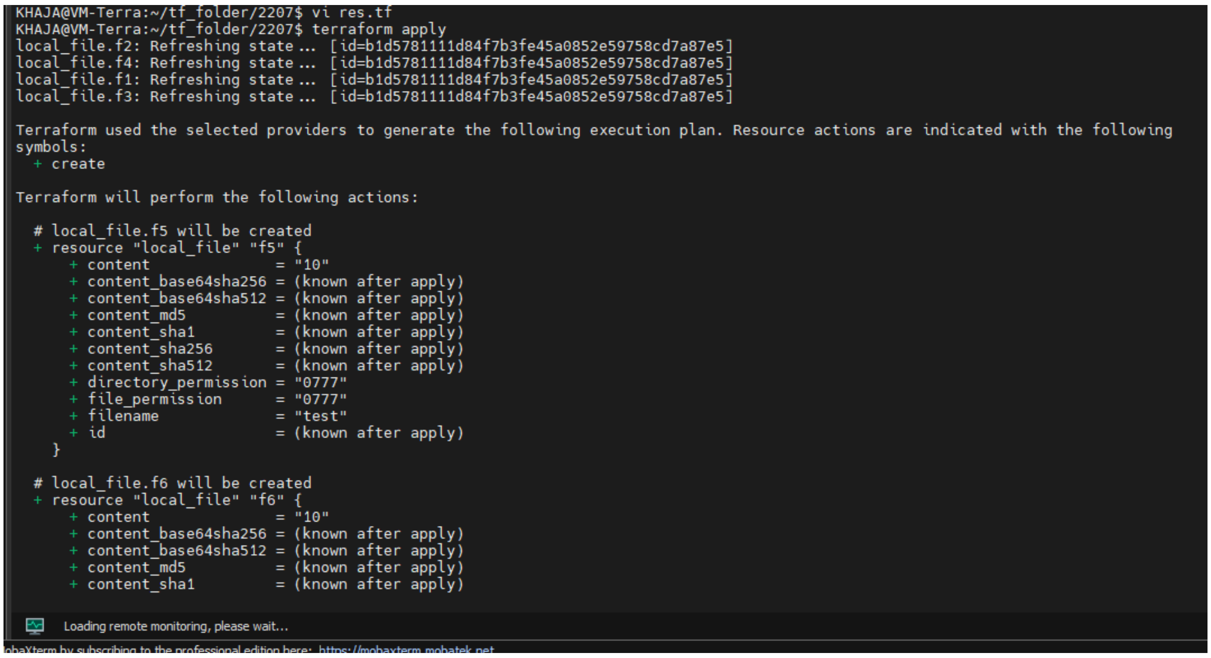
```
}
resource "local_file" "f5" {
  filename = var.filename5[0]
  content  = var.content
}
resource "local_file" "f6" {
  filename = var.filename6[2]
  content  = var.content
}
resource "local_file" "f7" {
  filename = var.filename7.name
  content  = var.content
}
resource "local_file" "f8" {
  filename = var.filename8.id
  content  = var.content
}

~
~
-- INSERT --
```

Know we have variable and resources with respect to their files



```
15  abc1.txt  abc2.txt  res.tf  terraform.tfstate  true  variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$ vi variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$ vi res.tf
KHAJA@VM-Terra:~/tf_folder/2207$
```

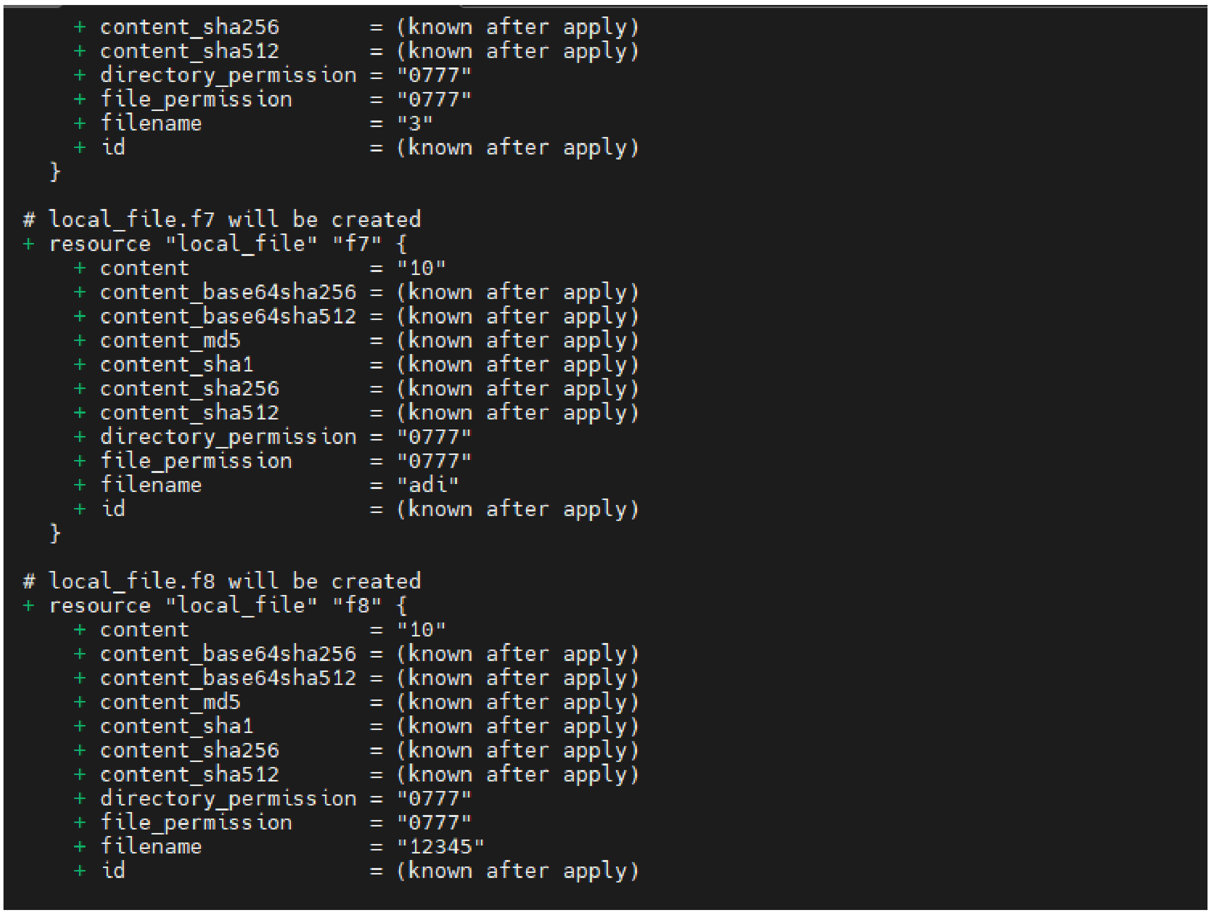
Execute the command apply



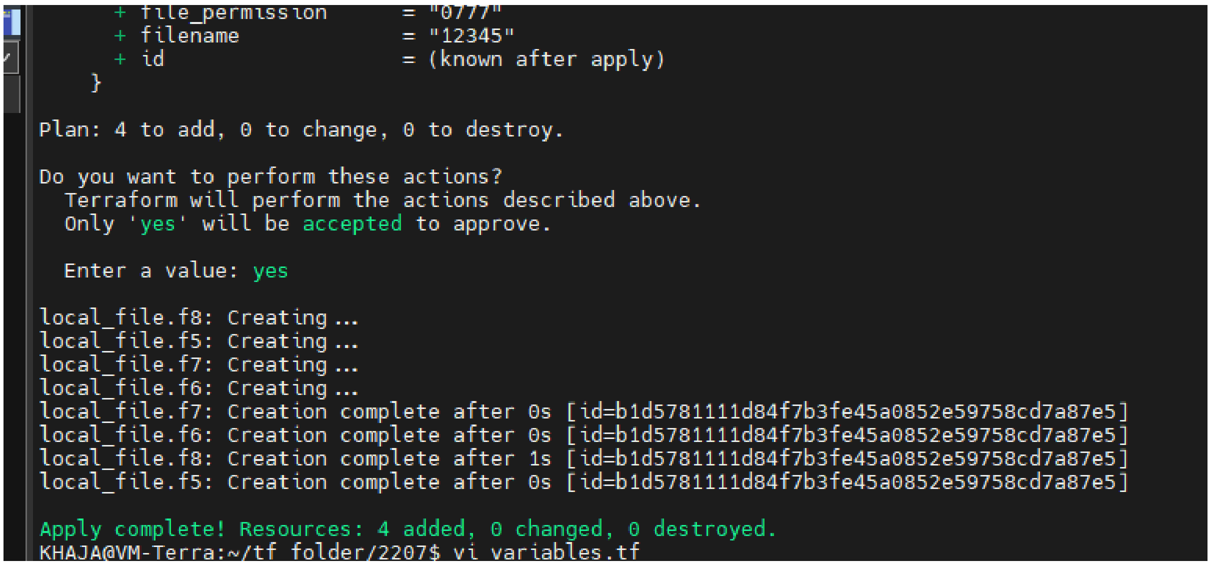
```
KHAJA@VM-Terra:~/tf_folder/2207$ vi res.tf
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f2: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.f5 will be created
  + resource "local_file" "f5" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "test"
      + id                   = (known after apply)
    }

  # local_file.f6 will be created
  + resource "local_file" "f6" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
```

Loading remote monitoring, please wait...

obaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



```
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "3"
      + id                   = (known after apply)
    }

  # local_file.f7 will be created
  + resource "local_file" "f7" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "adi"
      + id                   = (known after apply)
    }

  # local_file.f8 will be created
  + resource "local_file" "f8" {
      + content              = "10"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "12345"
      + id                   = (known after apply)
```



```
      + file_permission      = "0777"
      + filename             = "12345"
      + id                   = (known after apply)
    }

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.f8: Creating...
local_file.f5: Creating...
local_file.f7: Creating...
local_file.f6: Creating...
local_file.f7: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Creation complete after 1s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Creation complete after 0s [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf folder/2207$ vi variables.tf
```

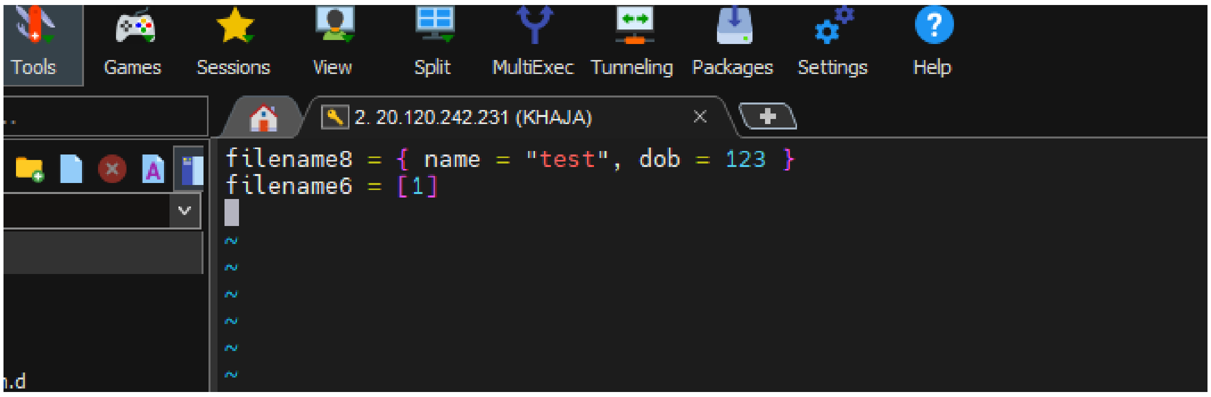
Inject the values by using .tfvar

vi terraform.tfvar

lets inject two values

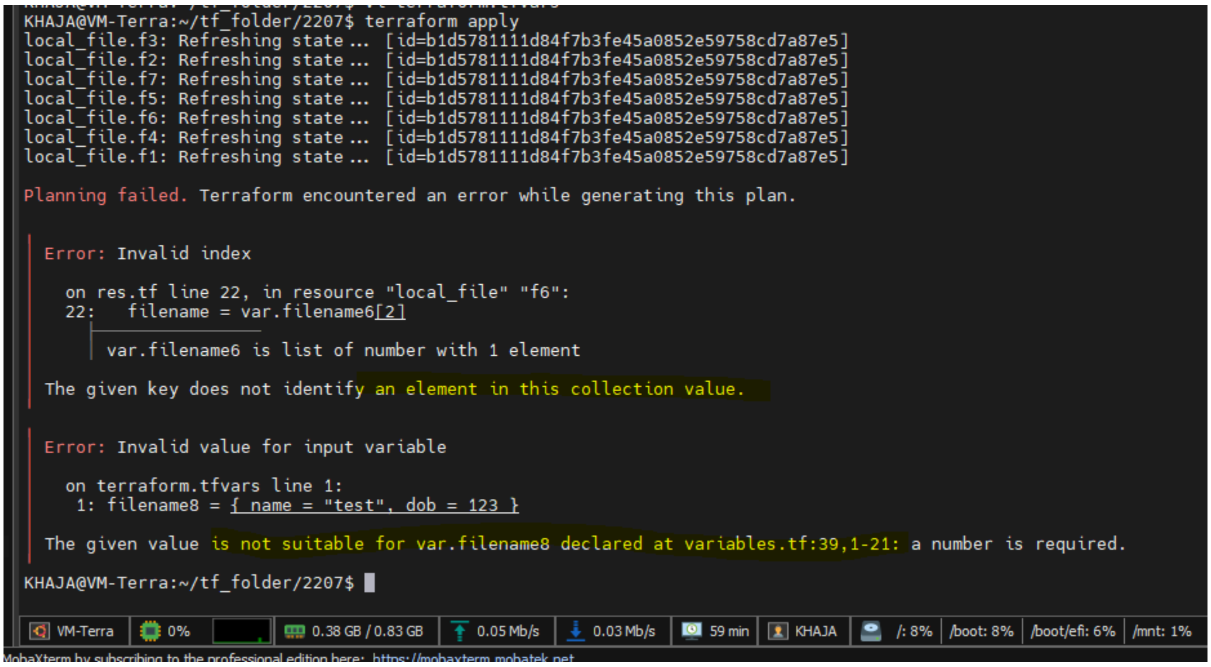
filename8 = { name = "test", dob = 123 }

filename6 = [1]   # we are injecting one value



```
filename8 = { name = "test", dob = 123 }
filename6 = [1]

~
~
~
~
~
~
```

When we execute terraform apply



```
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f3: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f2: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f7: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Planning failed. Terraform encountered an error while generating this plan.

│ Error: Invalid index
│
│   on res.tf line 22, in resource "local_file" "f6":
│   22:    filename = var.filename6[2]
│     ├────────────────
│     │ var.filename6 is list of number with 1 element
│
│ The given key does not identify an element in this collection value.

│ Error: Invalid value for input variable
│
│   on terraform.tfvars line 1:
│    1: filename8 = { name = "test", dob = 123 }
│
│ The given value is not suitable for var.filename8 declared at variables.tf:39,1-21: a number is required.

KHAJA@VM-Terra:~/tf_folder/2207$ █
```

Tuple :

A tuple in Terraform is a fixed-length, ordered collection of values where each position has its own specific type. Unlike lists (which must contain elements of the same type), tuples can contain elements of different types.

Tuple Characteristics:

- Fixed number of elements (length is part of the type definition)
- Each position has a specific type
- Defined using the syntax tuple([type1, type2, ...])

Example:

```
variable "example_tuple" {
  type = tuple([string, number, bool, list(string)])
  default = ["hello", 42, true, ["a", "b", "c"]]
}
```

Accessing Tuple Elements:

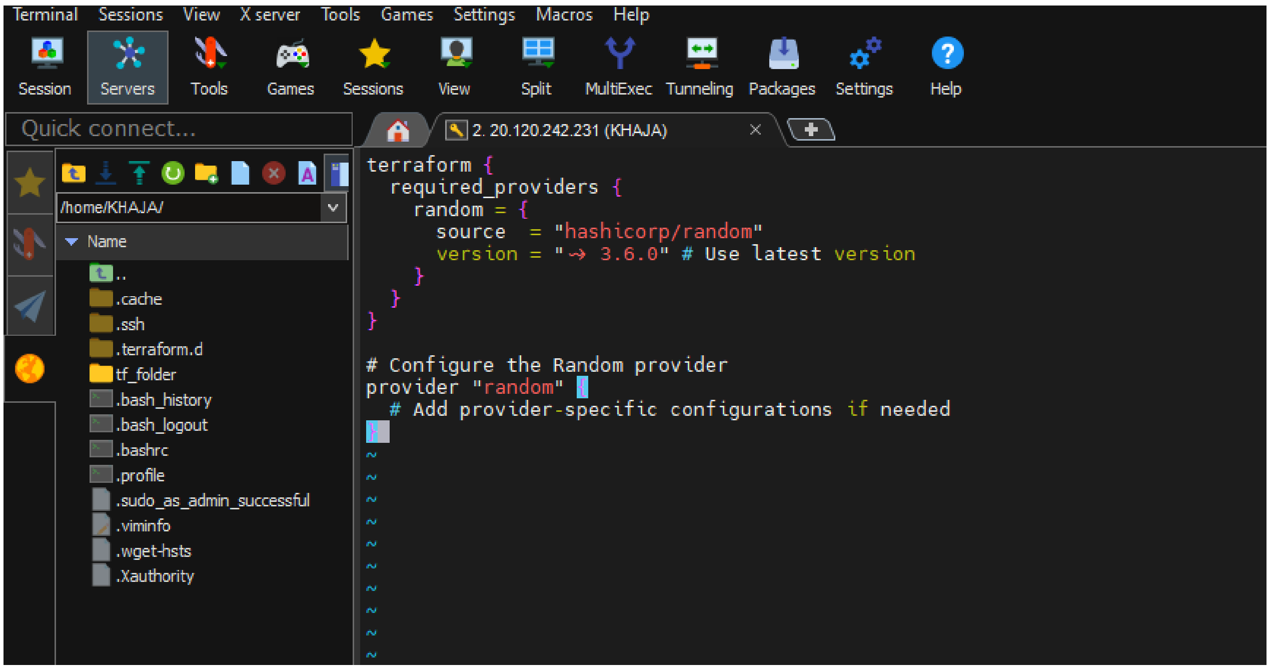
```
var.example_tuple[0]  # "hello" (string)
var.example_tuple[1]  # 42 (number)
var.example_tuple[2]  # true (bool)
var.example_tuple[3]  # ["a", "b", "c"] (list)
```

Random Provider

The **Random provider** generates random values during Terraform operations. It's useful for creating unique names, passwords, or other values that need to be unpredictable.

initialize the random provider:

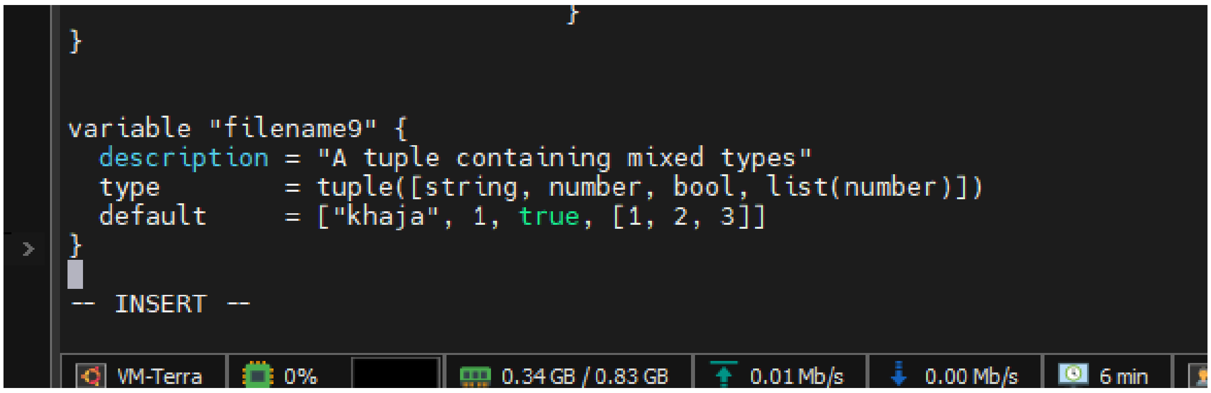
```
terraform {
 required_providers {
  random = {
   source = "hashicorp/random"
   version = "~> 3.0"
  }
 }
}
```


Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
Quick connect...
2. 20.120.242.231 (KHAJA)
/home/KHAJA/
Name
..
.cache
.ssh
.terraform.d
tf_folder
.bash_history
.bash_logout
.bashrc
.profile
.sudo_as_admin_successful
.viminfo
.wget-hsts
.Xauthority

```
terraform {
  required_providers {
    random = {
      source  = "hashicorp/random"
      version = "~> 3.6.0" # Use latest version
    }
  }
}

# Configure the Random provider
provider "random" {
  # Add provider-specific configurations if needed
}
```

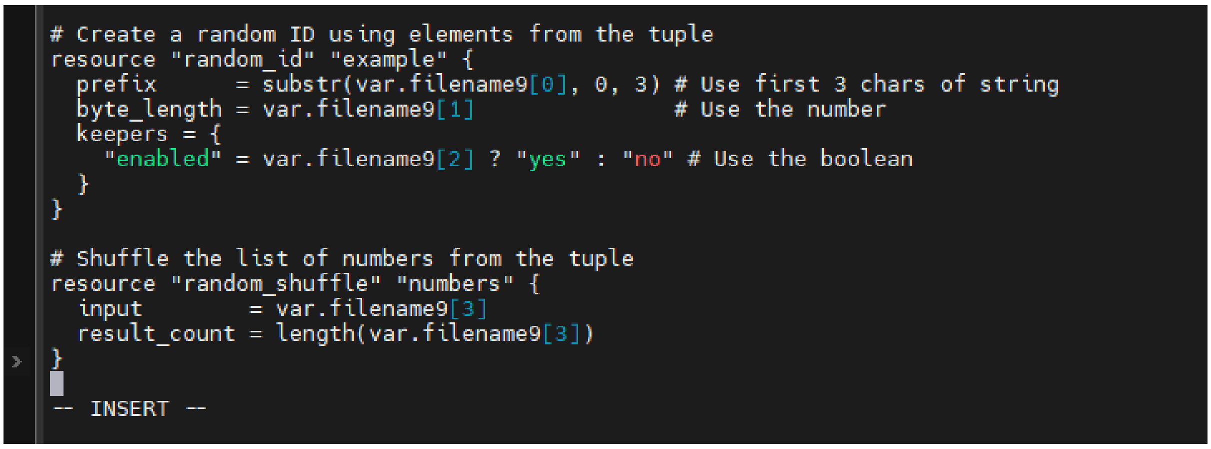
variables.tf - Declare of variables

```
variable "filename9" {
 description = "A tuple containing mixed types"
 type       = tuple([string, number, bool, list(number)])
 default     = ["khaja", 1, true, [1, 2, 3]]
}
```



```
}


variable "filename9" {
  description = "A tuple containing mixed types"
  type        = tuple([string, number, bool, list(number)])
  default     = ["khaja", 1, true, [1, 2, 3]]
}

-- INSERT --
```

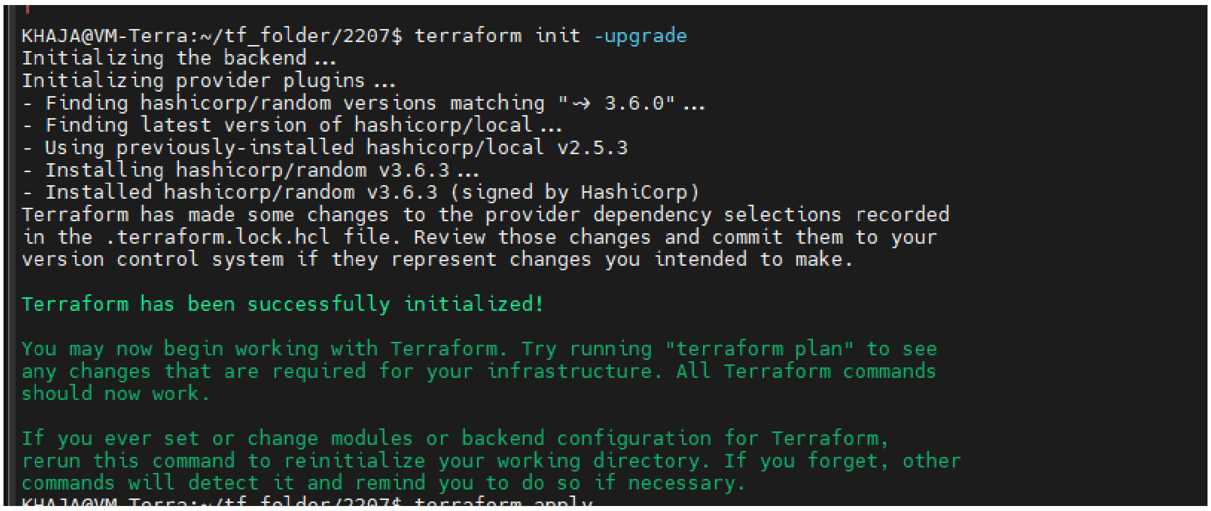
res.tf - Define resources using Random provider



```
# Create a random ID using elements from the tuple
resource "random_id" "example" {
  prefix      = substr(var.filename9[0], 0, 3) # Use first 3 chars of string
  byte_length = var.filename9[1]               # Use the number
  keepers = {
    "enabled" = var.filename9[2] ? "yes" : "no" # Use the boolean
  }
}

# Shuffle the list of numbers from the tuple
resource "random_shuffle" "numbers" {
  input        = var.filename9[3]
  result_count = length(var.filename9[3])
}

-- INSERT --
```

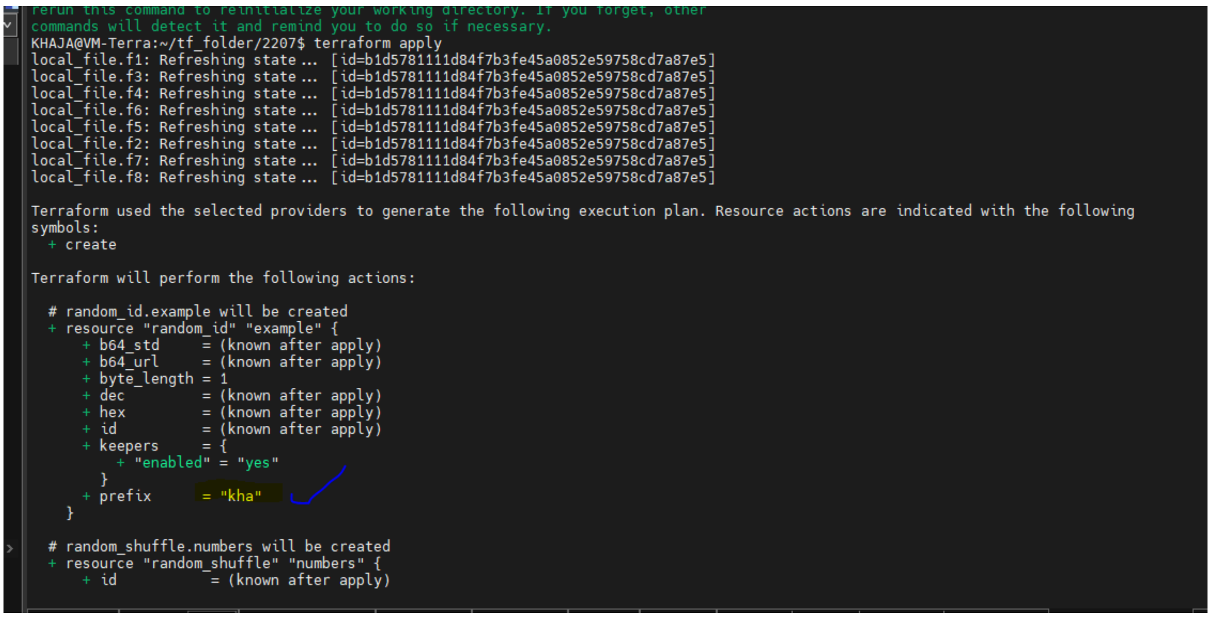
Added the resources to the existing res.tf

Execute the terraform init with update

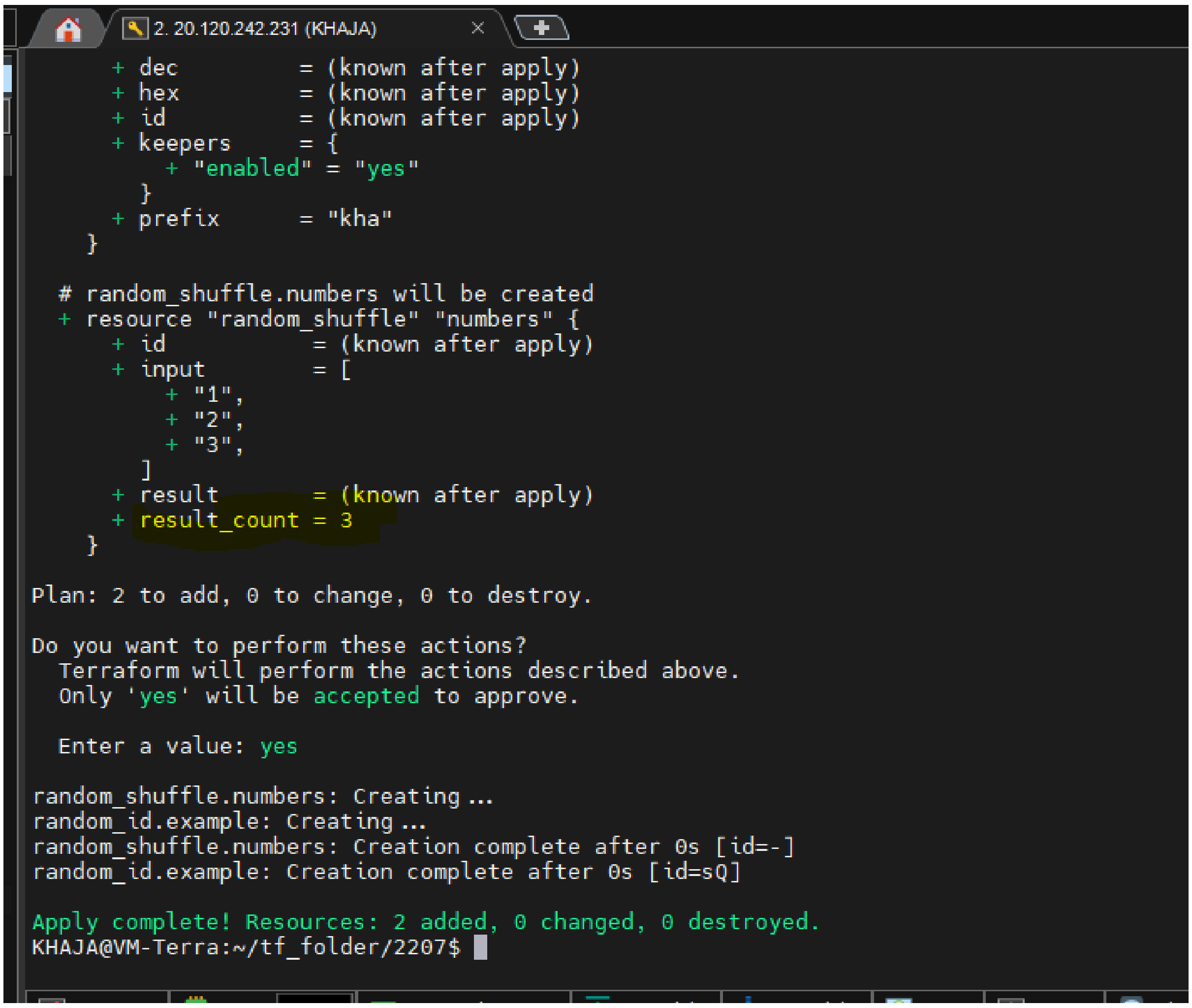


```
KHAJA@VM-Terra:~/tf_folder/2207$ terraform init -upgrade
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/random versions matching "~> 3.6.0"...
- Finding latest version of hashicorp/local...
- Using previously-installed hashicorp/local v2.5.3
- Installing hashicorp/random v3.6.3...
- Installed hashicorp/random v3.6.3 (signed by HashiCorp)
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
```

Use terraform apply to make the changes

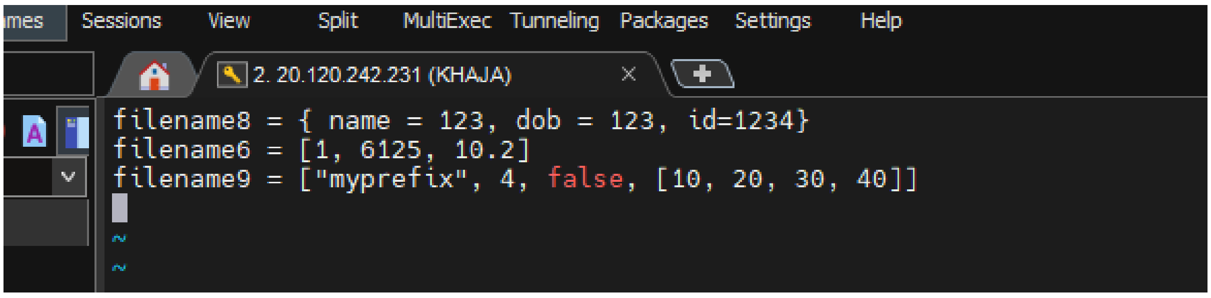


```
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f1: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f4: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f2: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f7: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # random_id.example will be created
  + resource "random_id" "example" {
      + b64_std     = (known after apply)
      + b64_url     = (known after apply)
      + byte_length = 1
      + dec         = (known after apply)
      + hex         = (known after apply)
      + id          = (known after apply)
      + keepers     = {
          + "enabled" = "yes"
        }
      + prefix      = "kha"
    }

  # random_shuffle.numbers will be created
  + resource "random_shuffle" "numbers" {
      + id           = (known after apply)
```

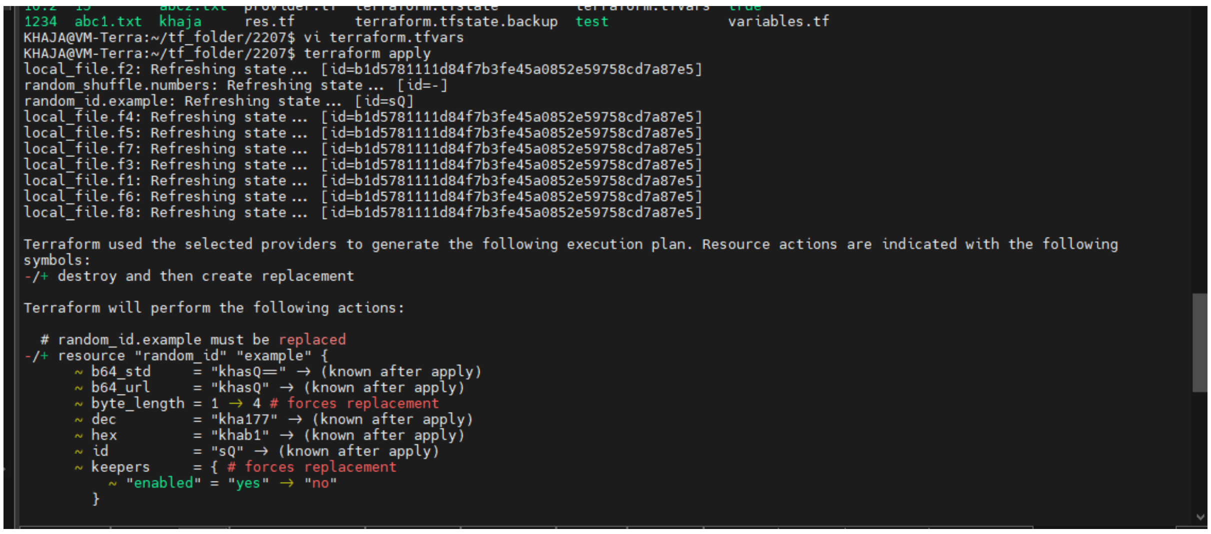


```
      + dec         = (known after apply)
      + hex         = (known after apply)
      + id          = (known after apply)
      + keepers     = {
          + "enabled" = "yes"
        }
      + prefix      = "kha"
    }

  # random_shuffle.numbers will be created
  + resource "random_shuffle" "numbers" {
      + id           = (known after apply)
      + input        = [
          + "1",
          + "2",
          + "3",
        ]
      + result       = (known after apply)
      + result_count = 3
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_shuffle.numbers: Creating ...
random_id.example: Creating ...
random_shuffle.numbers: Creation complete after 0s [id=-]
random_id.example: Creation complete after 0s [id=sQ]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$
```

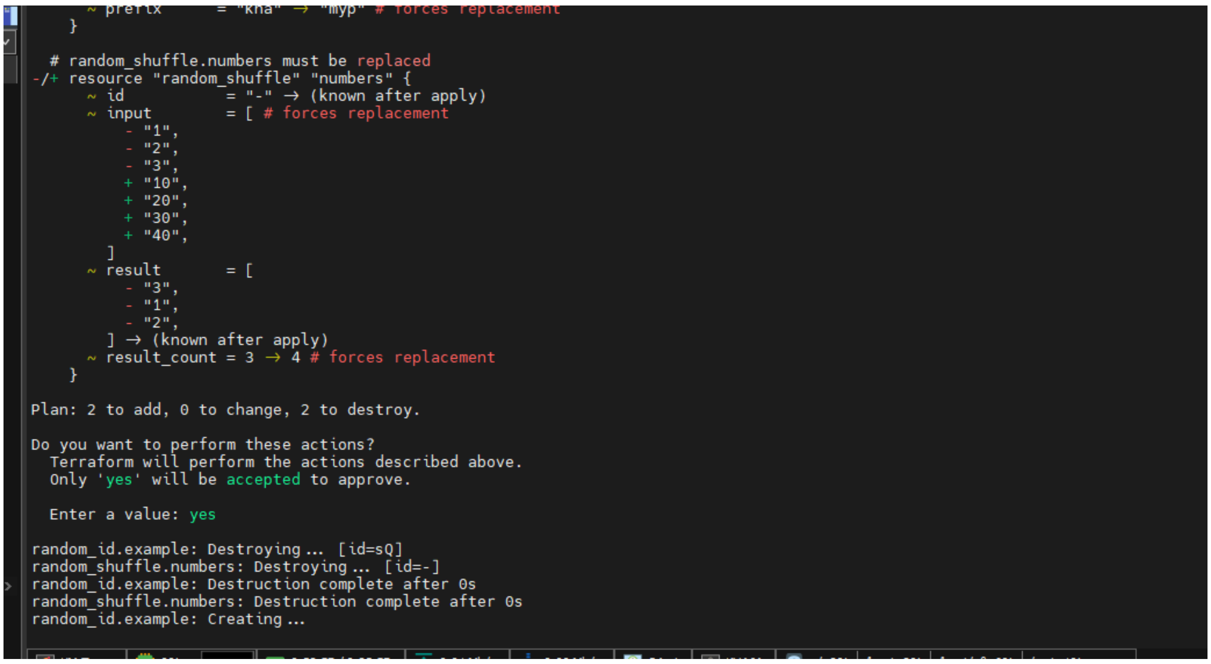
terraform.tfvars - Provide variable values (optional)



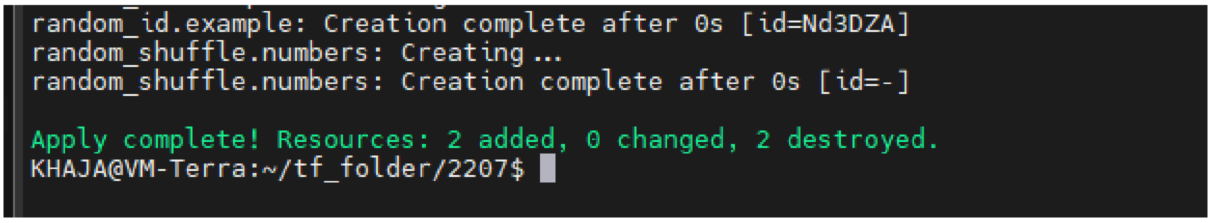
```
filename8 = { name = 123, dob = 123, id=1234}
filename6 = [1, 6125, 10.2]
filename9 = ["myprefix", 4, false, [10, 20, 30, 40]]

~
~
```



```
10.2  15      abc2.txt  provider.tf  terraform.tfstate         terraform.tfvars  true
1234  abc1.txt  khaja     res.tf       terraform.tfstate.backup  test              variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$ vi terraform.tfvars
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f2: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
random_shuffle.numbers: Refreshing state ... [id=-]
random_id.example: Refreshing state ... [id=sQ]
local_file.f4: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f7: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Refreshing state ... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # random_id.example must be replaced
-/+ resource "random_id" "example" {
      ~ b64_std     = "khasQ==" → (known after apply)
      ~ b64_url     = "khasQ" → (known after apply)
      ~ byte_length = 1 → 4 # forces replacement
      ~ dec         = "kha177" → (known after apply)
      ~ hex         = "khab1" → (known after apply)
      ~ id          = "sQ" → (known after apply)
      ~ keepers     = { # forces replacement
          ~ "enabled" = "yes" → "no"
        }
```



```
      ~ prefix      = "kha" → "myp" # forces replacement
    }

  # random_shuffle.numbers must be replaced
-/+ resource "random_shuffle" "numbers" {
      ~ id           = "-" → (known after apply)
      ~ input        = [ # forces replacement
          - "1",
          - "2",
          - "3",
          + "10",
          + "20",
          + "30",
          + "40",
        ]
      ~ result       = [
          - "3",
          - "1",
          - "2",
        ] → (known after apply)
      ~ result_count = 3 → 4 # forces replacement
    }

Plan: 2 to add, 0 to change, 2 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_id.example: Destroying ... [id=sQ]
random_shuffle.numbers: Destroying ... [id=-]
random_id.example: Destruction complete after 0s
random_shuffle.numbers: Destruction complete after 0s
random_id.example: Creating ...
```



```
random_id.example: Creation complete after 0s [id=Nd3DZA]
random_shuffle.numbers: Creating ...
random_shuffle.numbers: Creation complete after 0s [id=-]

Apply complete! Resources: 2 added, 0 changed, 2 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$
```

Key Benefits of This Structure:

1. **Separation of Concerns** - Each file has a clear purpose
2. **Better Maintainability** - Easier to find and modify components
3. **Reusability** - Provider configuration can be shared across modules
4. **Clear Dependencies** - Obvious where each component is defined

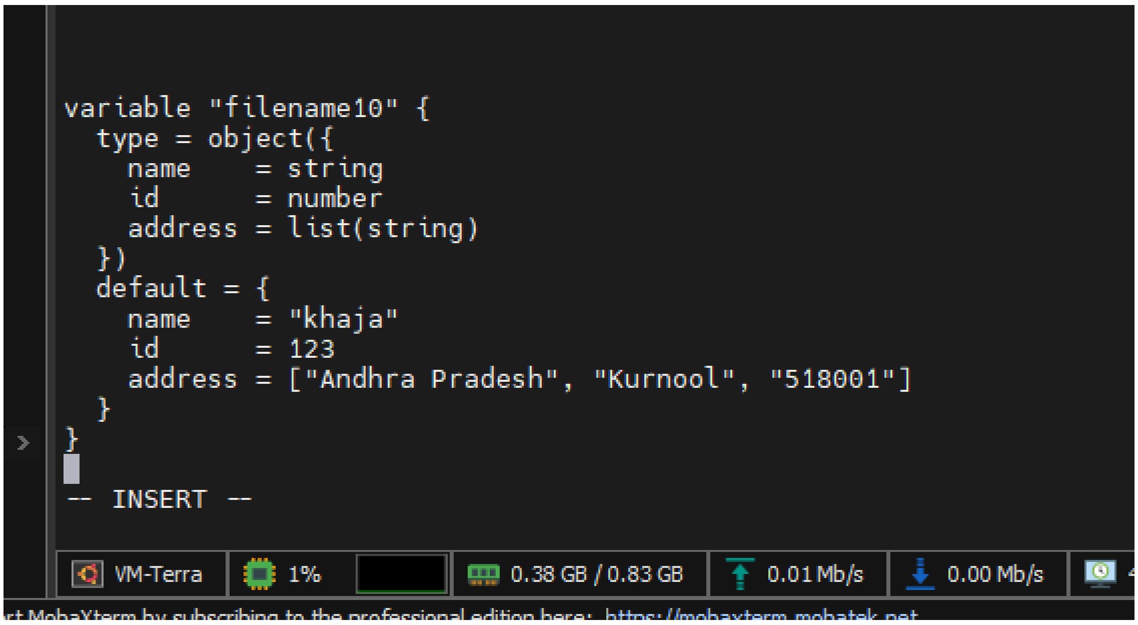
object :

user defind datatype -> wrapper on map, it declares the key structure and type

```
type = object({
        name = string
        id = number
        address = list(string)
})
default = {
        name = "adi"
        id = 123
        address = ["marathalli","bangalore","560037"]
}
```

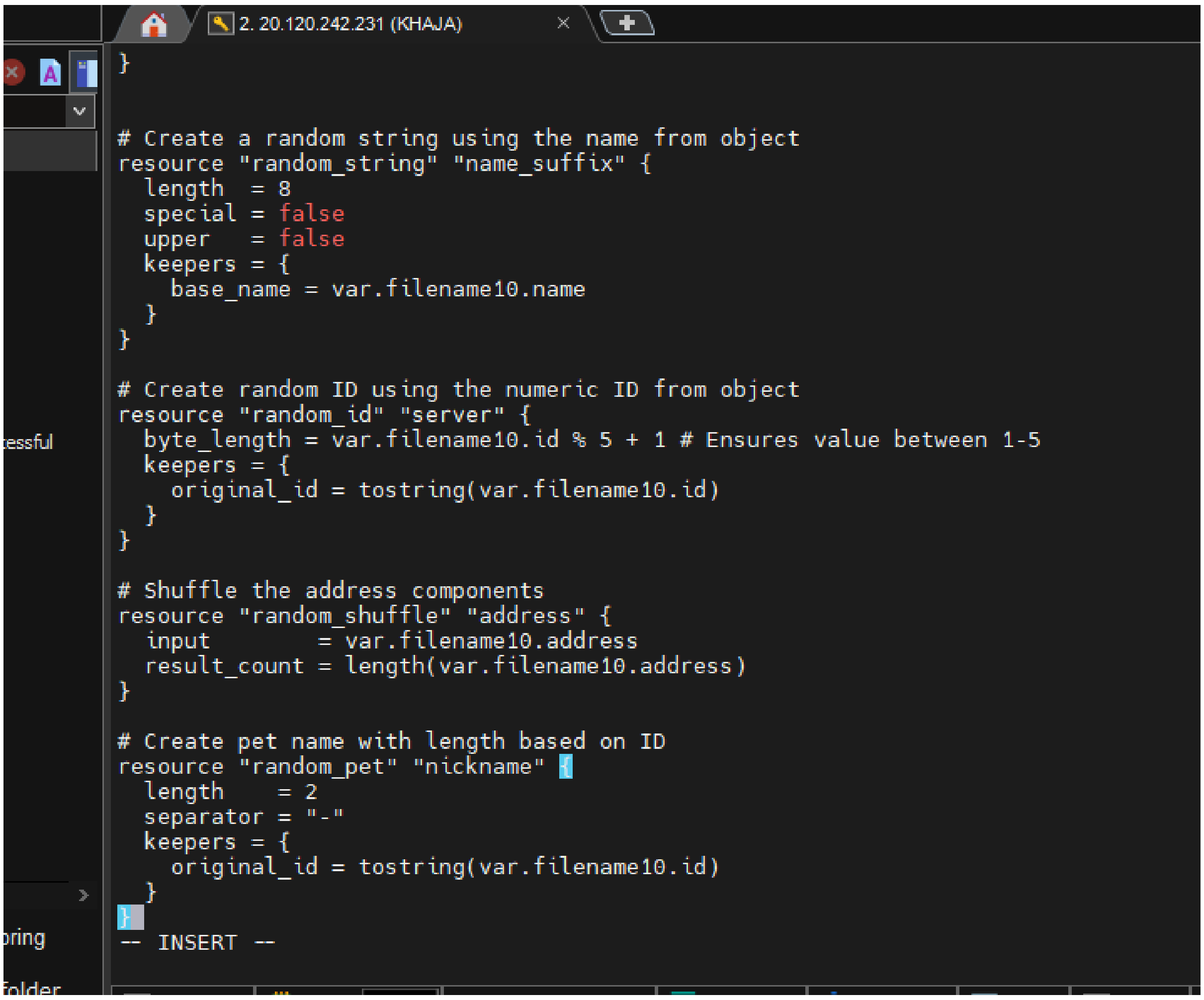
variables.tf - Object Variable Definition

```
variable "filename10" {
  type = object({
    name    = string
    id      = number
    address = list(string)
  })
  default = {
    name    = "khaja"
    id      = 123
    address = ["Andhra Pradesh", "Kurnool", "518001"]
  }
}
```



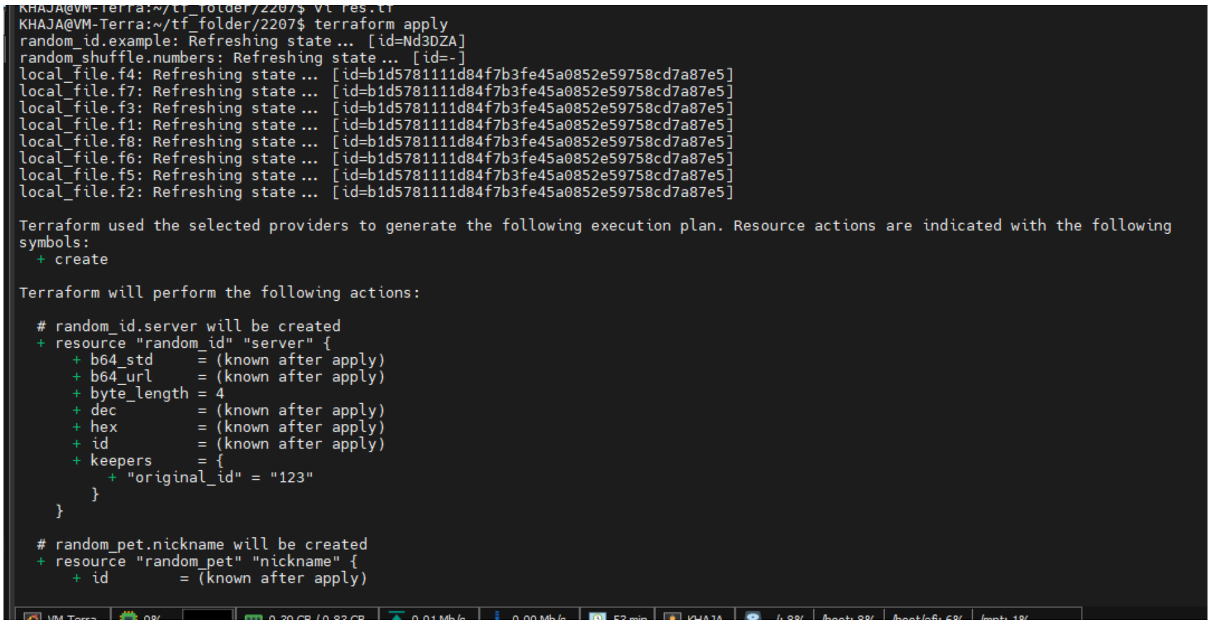
```
variable "filename10" {
  type = object({
    name    = string
    id      = number
    address = list(string)
  })
  default = {
    name    = "khaja"
    id      = 123
    address = ["Andhra Pradesh", "Kurnool", "518001"]
  }
}

-- INSERT --
```

res.tf - Random Provider Resources Using the Object

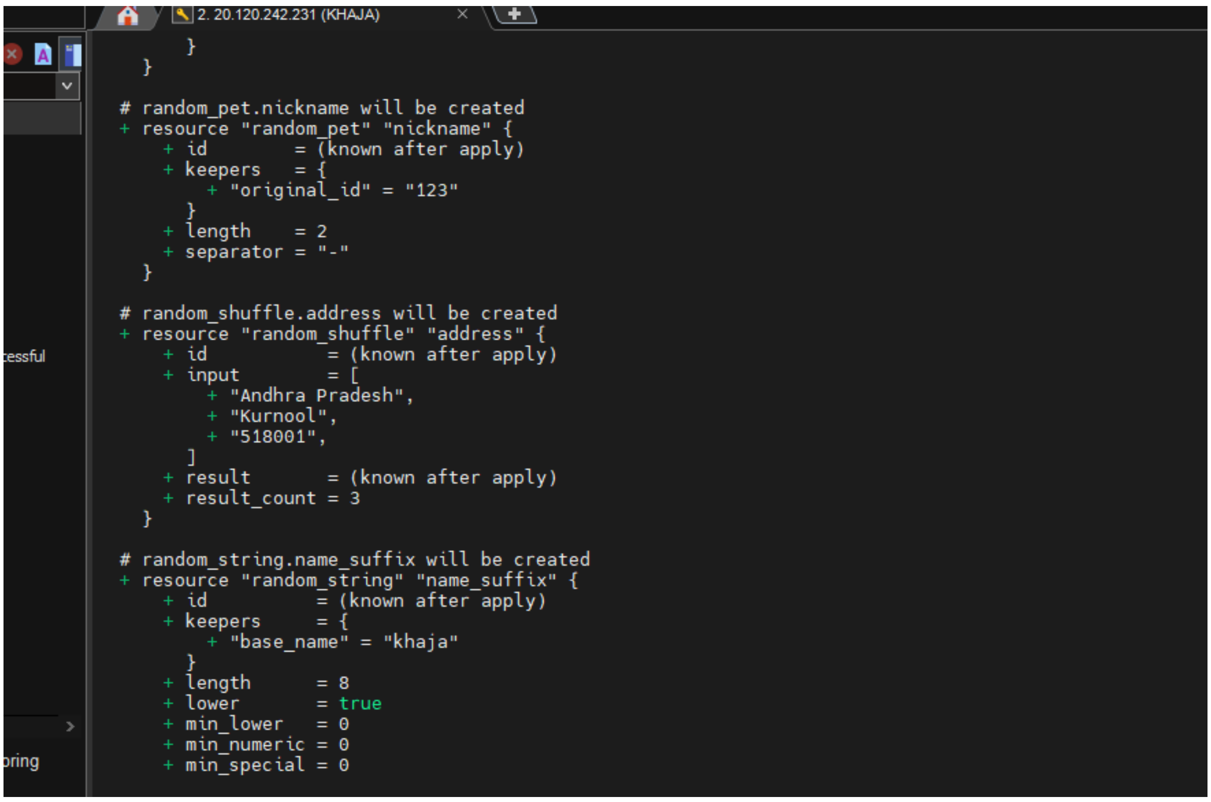


```
}

# Create a random string using the name from object
resource "random_string" "name_suffix" {
  length  = 8
  special = false
  upper   = false
  keepers = {
    base_name = var.filename10.name
  }
}

# Create random ID using the numeric ID from object
resource "random_id" "server" {
  byte_length = var.filename10.id % 5 + 1 # Ensures value between 1-5
  keepers = {
    original_id = tostring(var.filename10.id)
  }
}

# Shuffle the address components
resource "random_shuffle" "address" {
  input        = var.filename10.address
  result_count = length(var.filename10.address)
}

# Create pet name with length based on ID
resource "random_pet" "nickname" {
  length    = 2
  separator = "-"
  keepers = {
    original_id = tostring(var.filename10.id)
  }
}
-- INSERT --
```

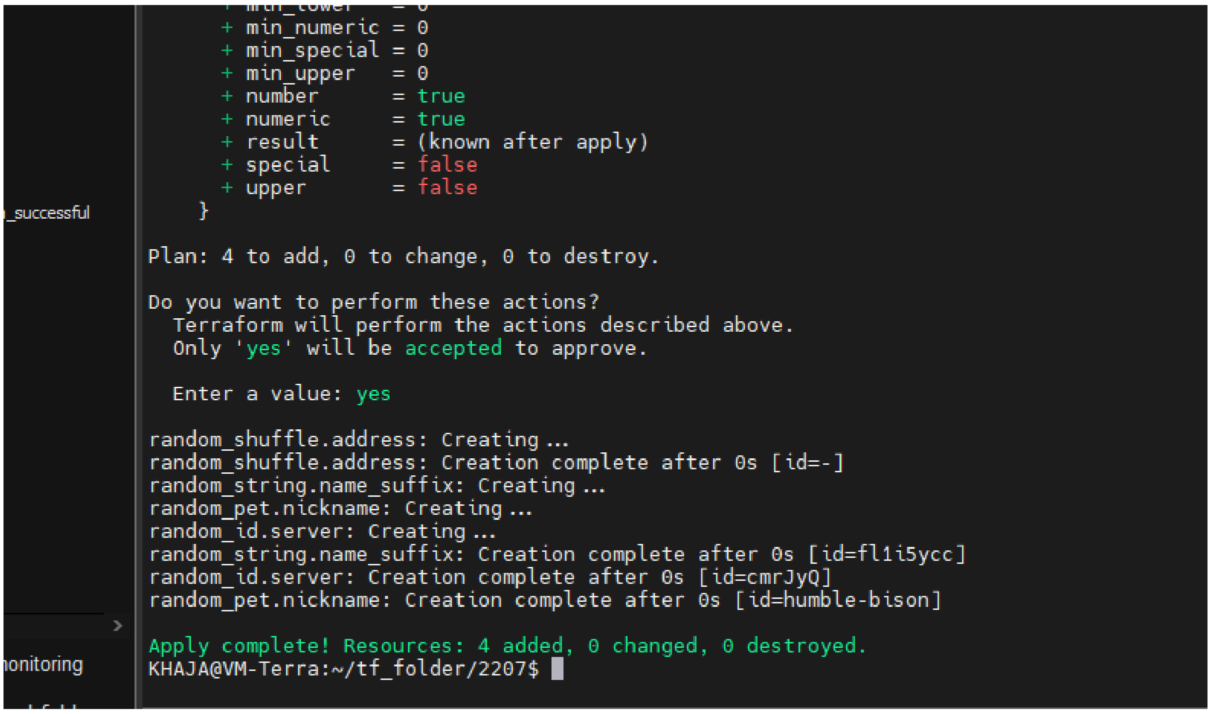
Execute the command terraform apply



```
KHAJA@VM-Terra:~/tf_folder/2207$ vi res.tf
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
random_id.example: Refreshing state... [id=Nd3DZA]
random_shuffle.numbers: Refreshing state... [id=-]
local_file.f4: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f7: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f2: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # random_id.server will be created
  + resource "random_id" "server" {
      + b64_std     = (known after apply)
      + b64_url     = (known after apply)
      + byte_length = 4
      + dec         = (known after apply)
      + hex         = (known after apply)
      + id          = (known after apply)
      + keepers     = {
          + "original_id" = "123"
        }
    }

  # random_pet.nickname will be created
  + resource "random_pet" "nickname" {
      + id        = (known after apply)
```

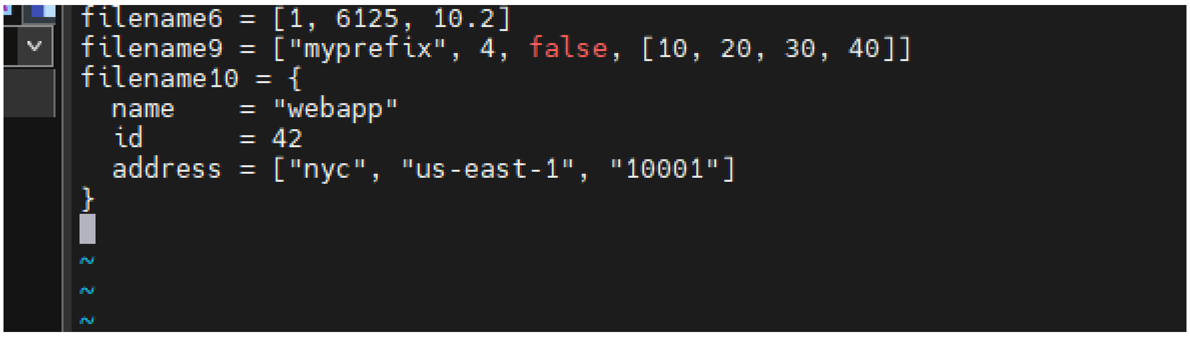


```
        }
    }

  # random_pet.nickname will be created
  + resource "random_pet" "nickname" {
      + id        = (known after apply)
      + keepers   = {
          + "original_id" = "123"
        }
      + length    = 2
      + separator = "-"
    }

  # random_shuffle.address will be created
  + resource "random_shuffle" "address" {
      + id           = (known after apply)
      + input        = [
          + "Andhra Pradesh",
          + "Kurnool",
          + "518001",
        ]
      + result       = (known after apply)
      + result_count = 3
    }

  # random_string.name_suffix will be created
  + resource "random_string" "name_suffix" {
      + id          = (known after apply)
      + keepers     = {
          + "base_name" = "khaja"
        }
      + length      = 8
      + lower       = true
      + min_lower   = 0
      + min_numeric = 0
      + min_special = 0
```



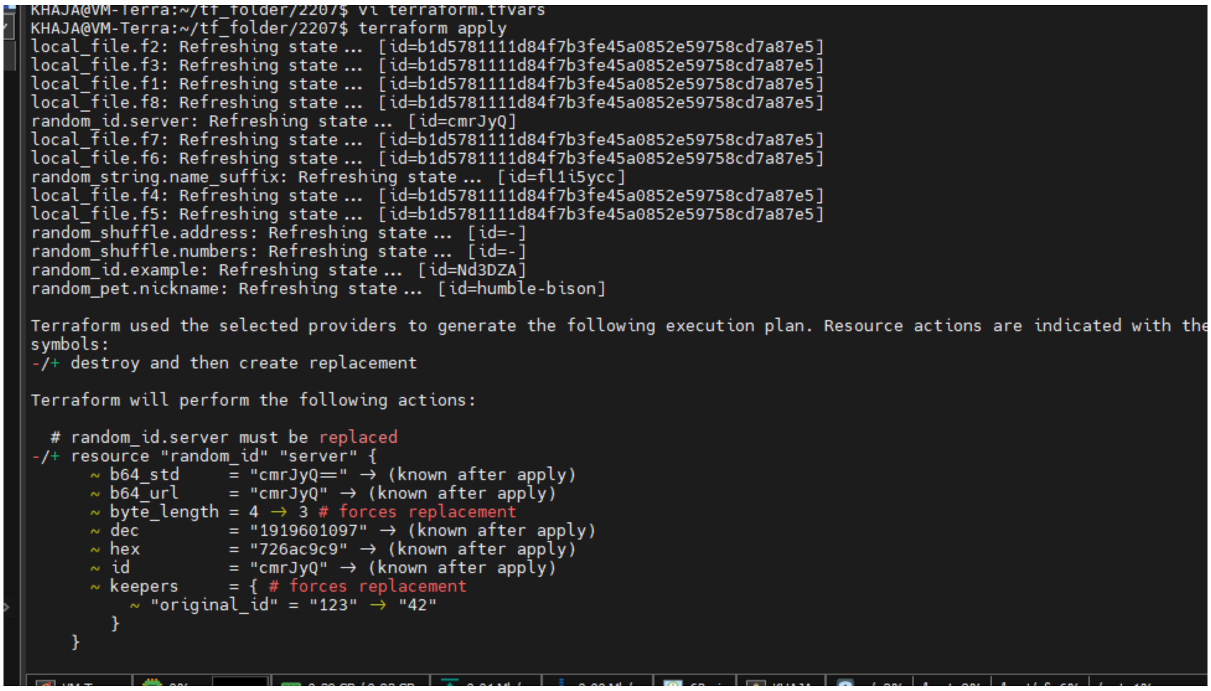
```
      + min_numeric = 0
      + min_special = 0
      + min_upper   = 0
      + number      = true
      + numeric     = true
      + result      = (known after apply)
      + special     = false
      + upper       = false
    }

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_shuffle.address: Creating...
random_shuffle.address: Creation complete after 0s [id=-]
random_string.name_suffix: Creating...
random_pet.nickname: Creating...
random_id.server: Creating...
random_string.name_suffix: Creation complete after 0s [id=fl1i5ycc]
random_id.server: Creation complete after 0s [id=cmrJyQ]
random_pet.nickname: Creation complete after 0s [id=humble-bison]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$
```

terraform.tfvars - Provide variable values (optional)

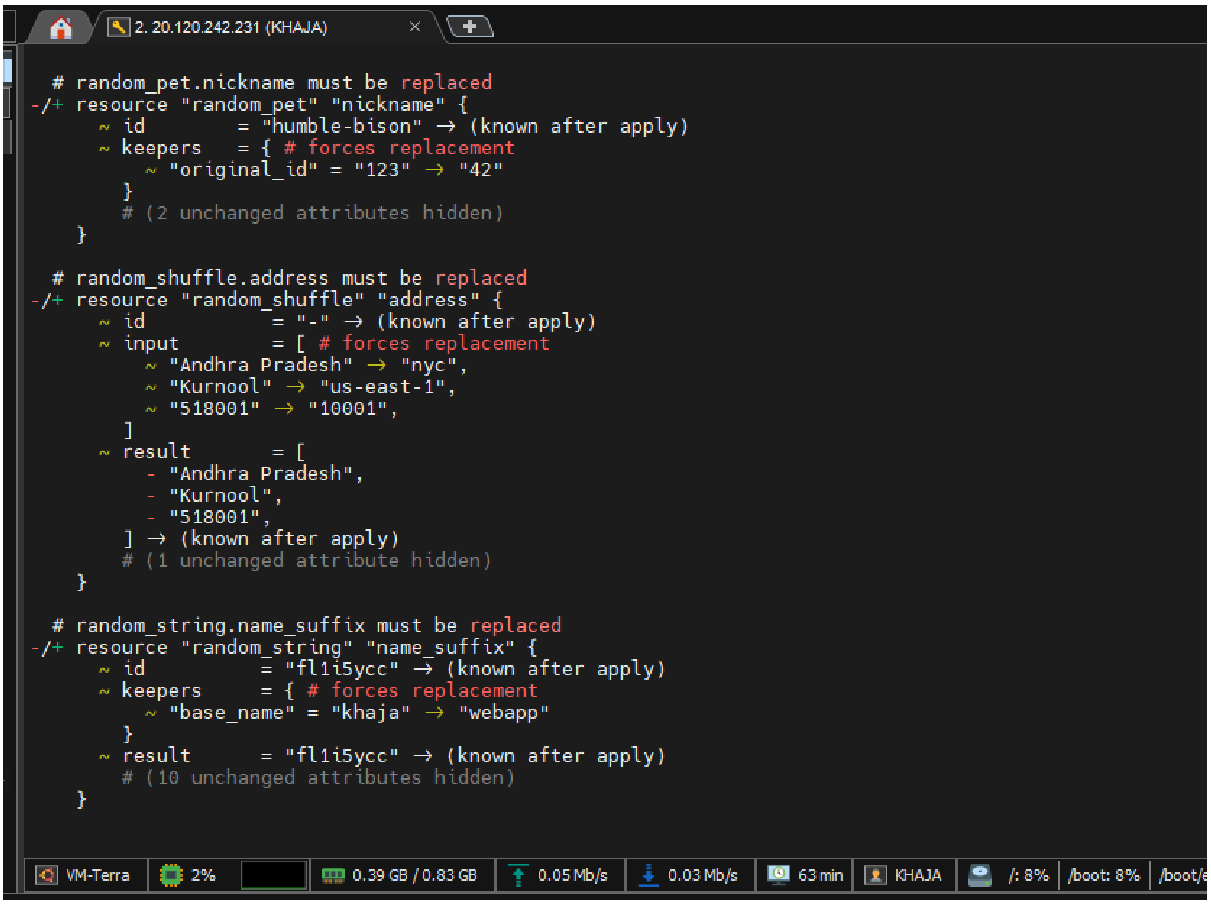


```
filename6 = [1, 6125, 10.2]
filename9 = ["myprefix", 4, false, [10, 20, 30, 40]]
filename10 = {
  name    = "webapp"
  id      = 42
  address = ["nyc", "us-east-1", "10001"]
}
~
~
~
```

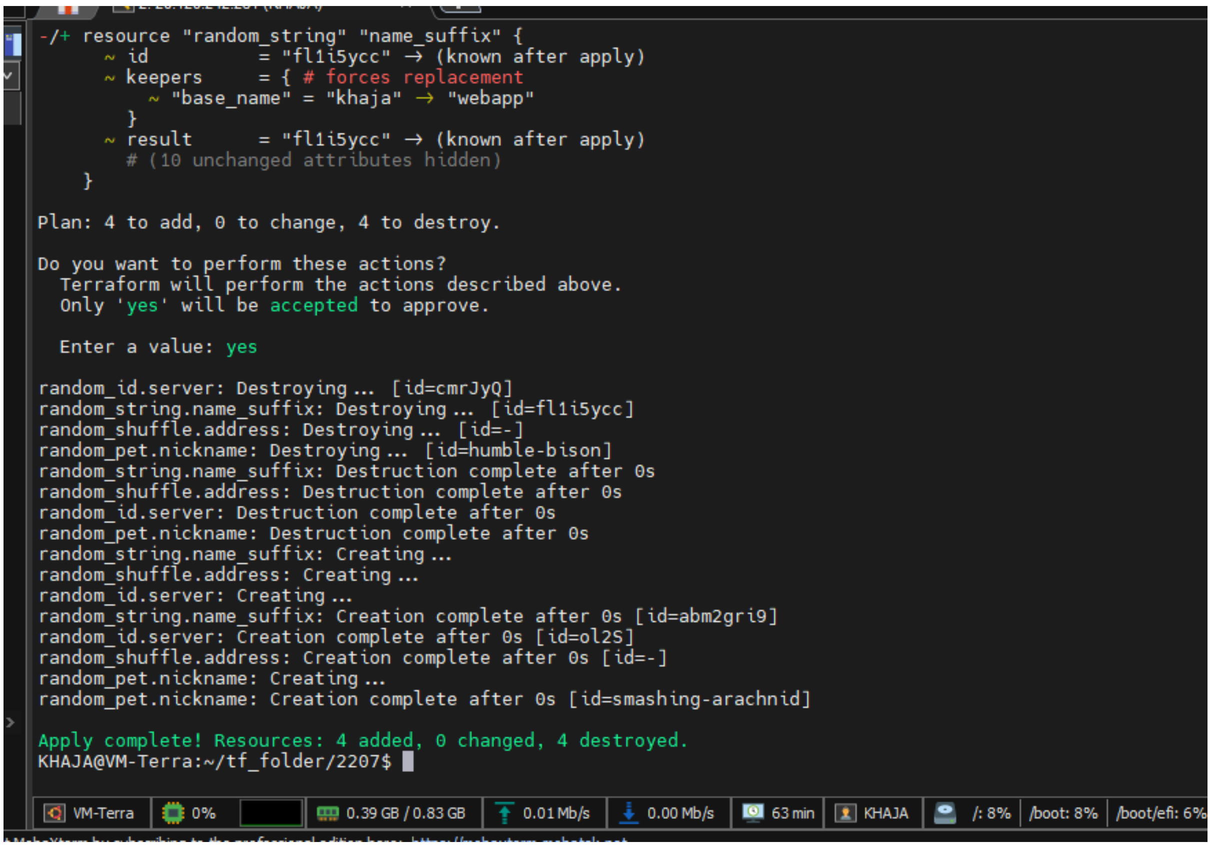
Again execute the terraform apply with the changes



```
KHAJA@VM-Terra:~/tf_folder/2207$ vi terraform.tfvars
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
local_file.f2: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f3: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f1: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f8: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
random_id.server: Refreshing state... [id=cmrJyQ]
local_file.f7: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f6: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
random_string.name_suffix: Refreshing state... [id=fl1i5ycc]
local_file.f4: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
local_file.f5: Refreshing state... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
random_shuffle.address: Refreshing state... [id=-]
random_shuffle.numbers: Refreshing state... [id=-]
random_id.example: Refreshing state... [id=Nd3DZA]
random_pet.nickname: Refreshing state... [id=humble-bison]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # random_id.server must be replaced
-/+ resource "random_id" "server" {
      ~ b64_std     = "cmrJyQ==" -> (known after apply)
      ~ b64_url     = "cmrJyQ" -> (known after apply)
      ~ byte_length = 4 -> 3 # forces replacement
      ~ dec         = "1919601097" -> (known after apply)
      ~ hex         = "726ac9c9" -> (known after apply)
      ~ id          = "cmrJyQ" -> (known after apply)
      ~ keepers     = { # forces replacement
          ~ "original_id" = "123" -> "42"
        }
    }
```



```
  # random_pet.nickname must be replaced
-/+ resource "random_pet" "nickname" {
      ~ id         = "humble-bison" -> (known after apply)
      ~ keepers    = { # forces replacement
          ~ "original_id" = "123" -> "42"
        }
        # (2 unchanged attributes hidden)
    }

  # random_shuffle.address must be replaced
-/+ resource "random_shuffle" "address" {
      ~ id            = "-" -> (known after apply)
      ~ input         = [ # forces replacement
          ~ "Andhra Pradesh" -> "nyc",
          ~ "Kurnool" -> "us-east-1",
          ~ "518001" -> "10001",
        ]
      ~ result        = [
          - "Andhra Pradesh",
          - "Kurnool",
          - "518001",
        ] -> (known after apply)
        # (1 unchanged attribute hidden)
    }

  # random_string.name_suffix must be replaced
-/+ resource "random_string" "name_suffix" {
      ~ id          = "fl1i5ycc" -> (known after apply)
      ~ keepers     = { # forces replacement
          ~ "base_name" = "khaja" -> "webapp"
        }
      ~ result      = "fl1i5ycc" -> (known after apply)
        # (10 unchanged attributes hidden)
    }
```



```
-/+ resource "random_string" "name_suffix" {
      ~ id          = "fl1i5ycc" -> (known after apply)
      ~ keepers     = { # forces replacement
          ~ "base_name" = "khaja" -> "webapp"
        }
      ~ result      = "fl1i5ycc" -> (known after apply)
        # (10 unchanged attributes hidden)
    }

Plan: 4 to add, 0 to change, 4 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_id.server: Destroying... [id=cmrJyQ]
random_string.name_suffix: Destroying... [id=fl1i5ycc]
random_shuffle.address: Destroying... [id=-]
random_pet.nickname: Destroying... [id=humble-bison]
random_string.name_suffix: Destruction complete after 0s
random_shuffle.address: Destruction complete after 0s
random_id.server: Destruction complete after 0s
random_pet.nickname: Destruction complete after 0s
random_string.name_suffix: Creating...
random_shuffle.address: Creating...
random_id.server: Creating...
random_string.name_suffix: Creation complete after 0s [id=abm2gri9]
random_id.server: Creation complete after 0s [id=ol2S]
random_shuffle.address: Creation complete after 0s [id=-]
random_pet.nickname: Creating...
random_pet.nickname: Creation complete after 0s [id=smashing-arachnid]

Apply complete! Resources: 4 added, 0 changed, 4 destroyed.
KHAJA@VM-Terra:~/tf_folder/2207$ █
```

Null provider

A provider for creating "no-op" resources that don't provision real infrastructure.
Primary Use Cases:

- Dependency management
- Debugging variables
- Triggering side effects

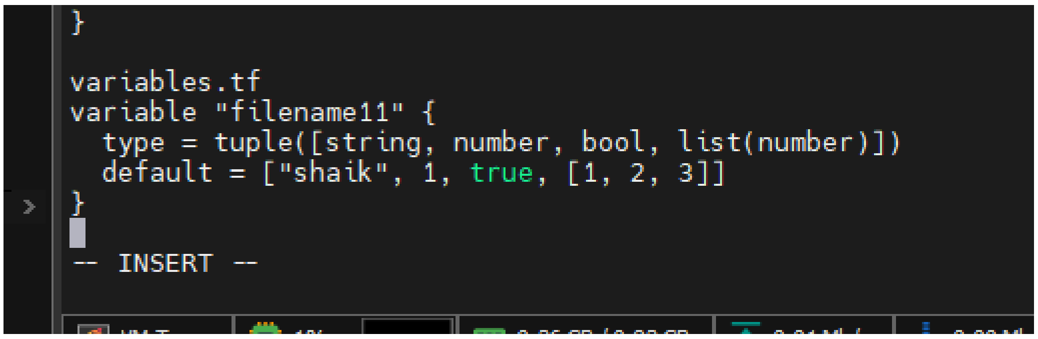
Example:

```
resource "null_resource" "debug" {
  triggers = {
    timestamp = timestamp()  # Re-runs when this changes
  }
  provisioner "local-exec" {
    command = "echo 'Triggered at ${self.triggers.timestamp}'"
  }
}
```

Using Null Provider with Tuple

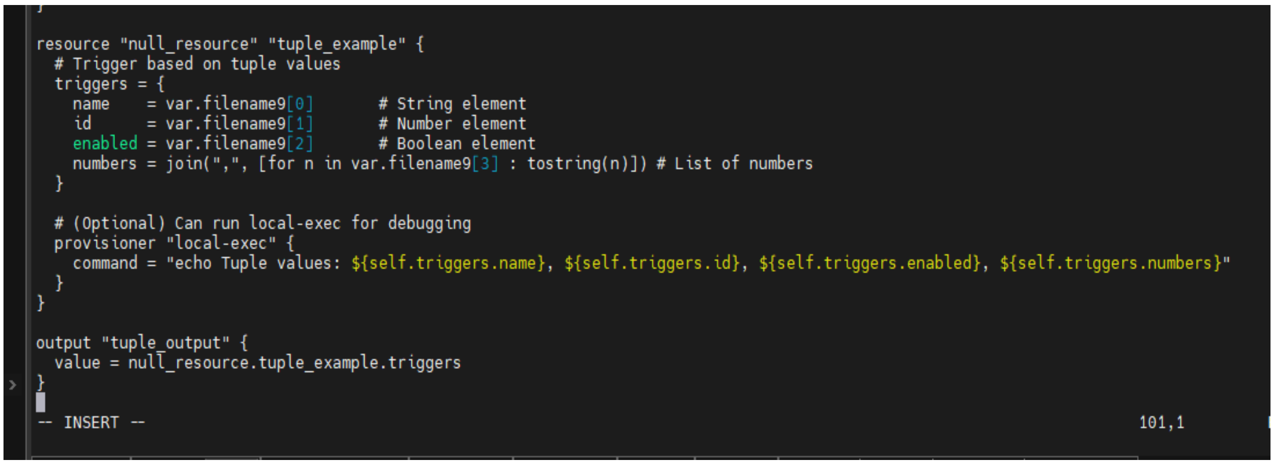
variables.tf

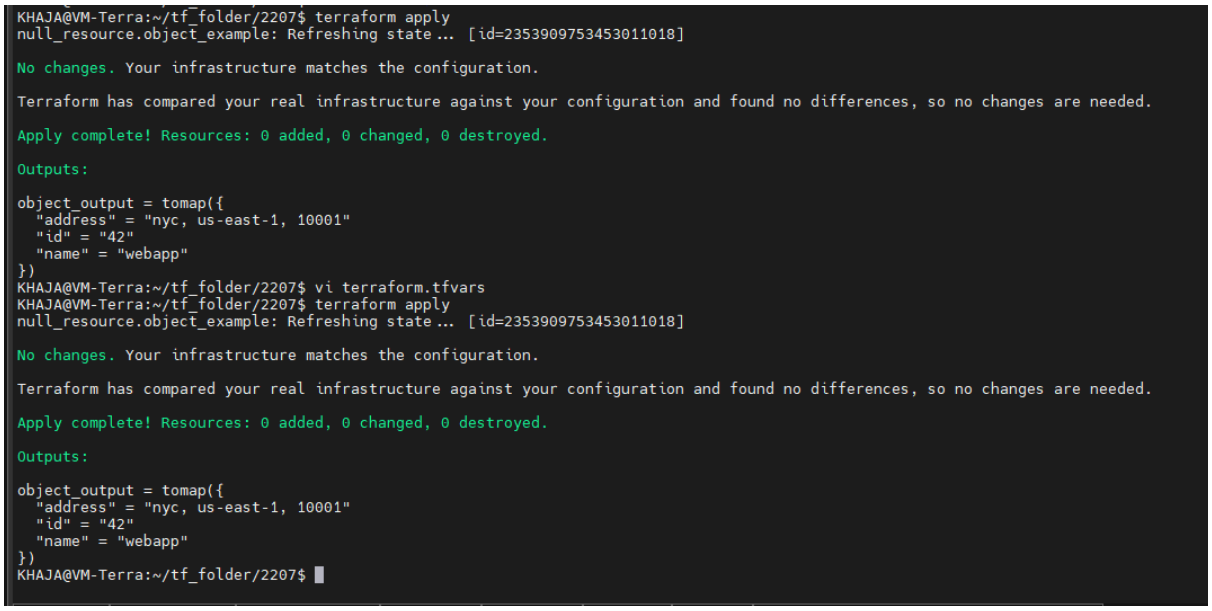
```
variable "filename11" {
  type = tuple([string, number, bool, list(number)])
  default = ["shaik", 1, true, [1, 2, 3]]
}
```



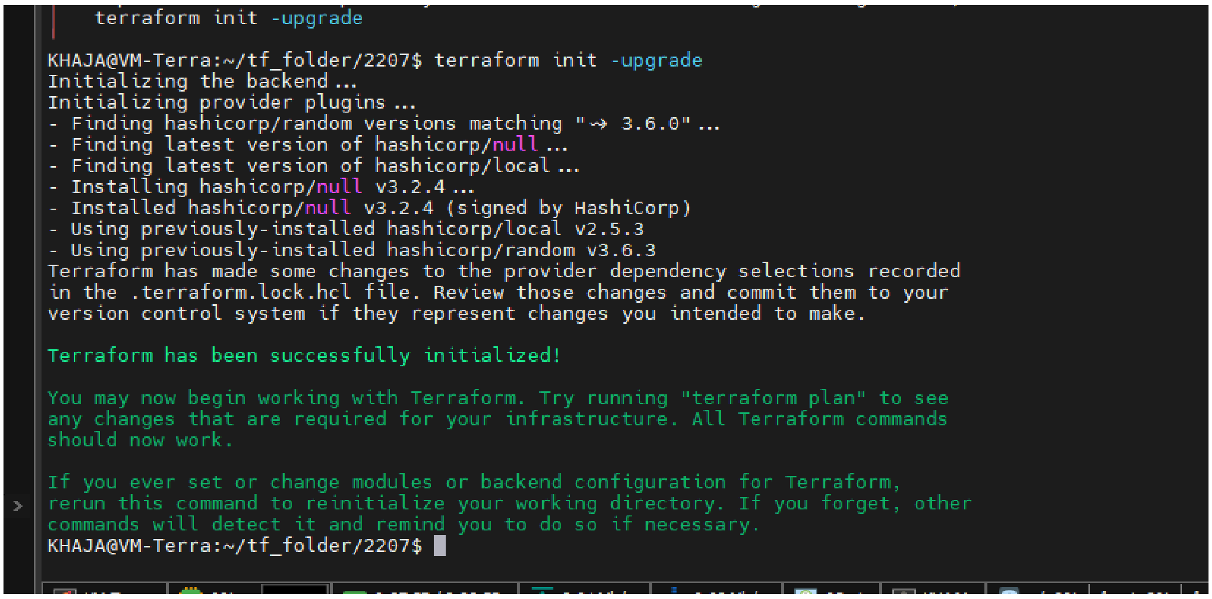
```
}

variables.tf
variable "filename11" {
  type = tuple([string, number, bool, list(number)])
  default = ["shaik", 1, true, [1, 2, 3]]
}

-- INSERT --
```

res.tf for declaring reources


```
resource "null_resource" "tuple_example" {
  # Trigger based on tuple values
  triggers = {
    name    = var.filename9[0]      # String element
    id      = var.filename9[1]      # Number element
    enabled = var.filename9[2]      # Boolean element
    numbers = join(",", [for n in var.filename9[3] : tostring(n)]) # List of numbers
  }

  # (Optional) Can run local-exec for debugging
  provisioner "local-exec" {
    command = "echo Tuple values: ${self.triggers.name}, ${self.triggers.id}, ${self.triggers.enabled}, ${self.triggers.numbers}"
  }
}

output "tuple_output" {
  value = null_resource.tuple_example.triggers
}

-- INSERT --                                                                                   101,1
```

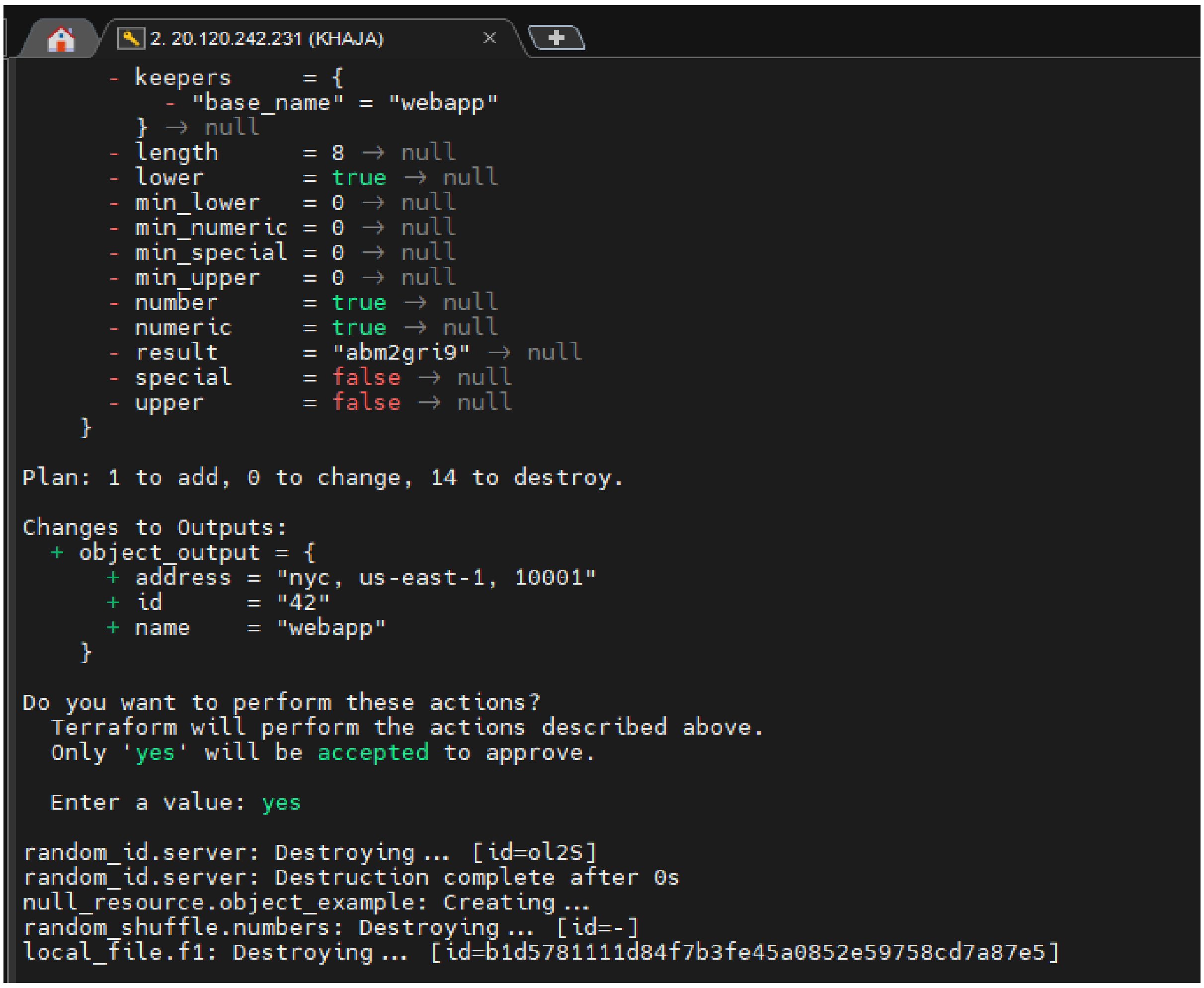

```
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
null_resource.object_example: Refreshing state... [id=2353909753453011018]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

object_output = tomap({
  "address" = "nyc, us-east-1, 10001"
  "id" = "42"
  "name" = "webapp"
})
KHAJA@VM-Terra:~/tf_folder/2207$ vi terraform.tfvars
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
null_resource.object_example: Refreshing state... [id=2353909753453011018]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

object_output = tomap({
  "address" = "nyc, us-east-1, 10001"
  "id" = "42"
  "name" = "webapp"
})
KHAJA@VM-Terra:~/tf_folder/2207$
```

Execute the command terraform init



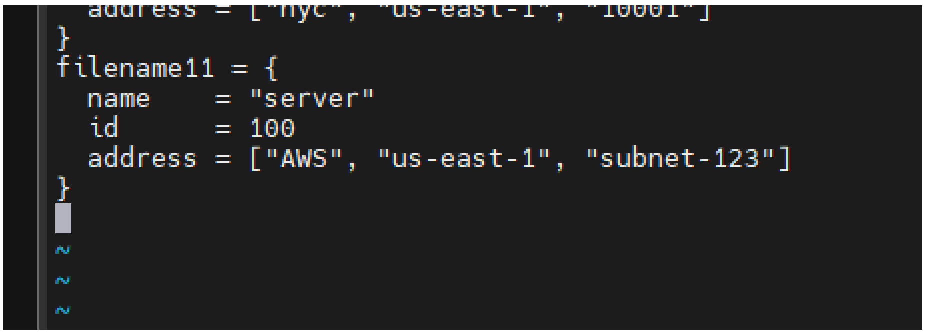
```
    terraform init -upgrade

KHAJA@VM-Terra:~/tf_folder/2207$ terraform init -upgrade
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/random versions matching "~> 3.6.0"...
- Finding latest version of hashicorp/null...
- Finding latest version of hashicorp/local...
- Installing hashicorp/null v3.2.4...
- Installed hashicorp/null v3.2.4 (signed by HashiCorp)
- Using previously-installed hashicorp/local v2.5.3
- Using previously-installed hashicorp/random v3.6.3
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
KHAJA@VM-Terra:~/tf_folder/2207$
```

Execute the command terraform apply



```
      - keepers     = {
          - "base_name" = "webapp"
        } -> null
      - length      = 8 -> null
      - lower       = true -> null
      - min_lower   = 0 -> null
      - min_numeric = 0 -> null
      - min_special = 0 -> null
      - min_upper   = 0 -> null
      - number      = true -> null
      - numeric     = true -> null
      - result      = "abm2gri9" -> null
      - special     = false -> null
      - upper       = false -> null
    }

Plan: 1 to add, 0 to change, 14 to destroy.

Changes to Outputs:
  + object_output = {
      + address = "nyc, us-east-1, 10001"
      + id      = "42"
      + name    = "webapp"
    }

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

random_id.server: Destroying... [id=ol2S]
random_id.server: Destruction complete after 0s
null_resource.object_example: Creating...
random_shuffle.numbers: Destroying... [id=-]
local_file.f1: Destroying... [id=b1d5781111d84f7b3fe45a0852e59758cd7a87e5]
```

terraform.tfvars (Optional)

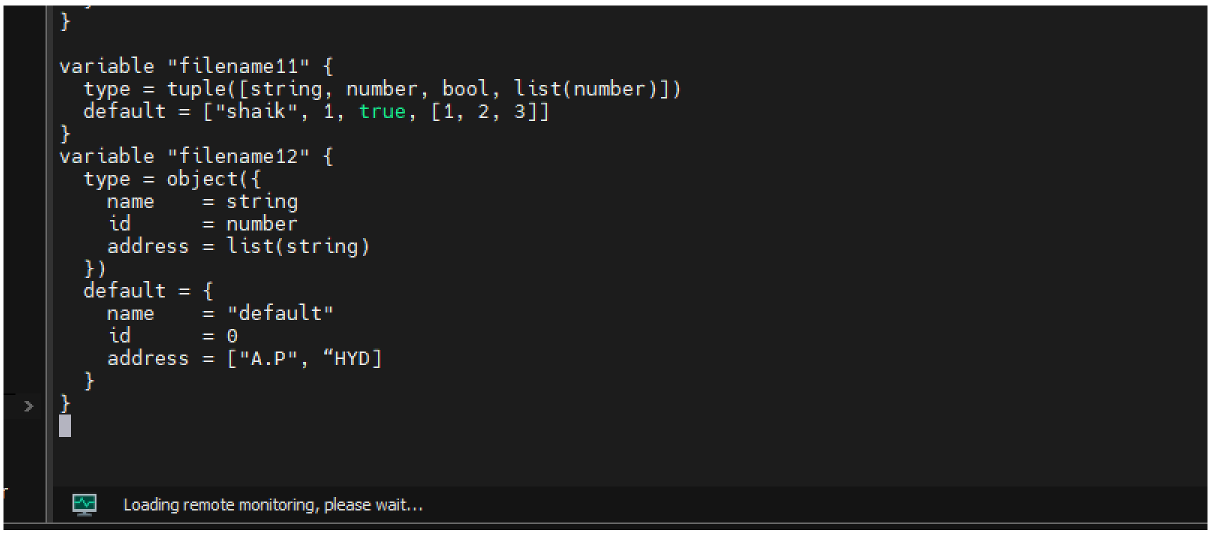


```
  address = ["nyc", "us-east-1", "10001"]
}
filename11 = {
  name    = "server"
  id      = 100
  address = ["AWS", "us-east-1", "subnet-123"]
}
~
~
~
```

Using Null Provider with Object

variables.tf

```
variable "filename12" {
  type = object({
    name    = string
    id      = number
    address = list(string)
  })
  default = {
    name    = "default"
    id      = 0
    address = ["A.P", “HYD]
  }
}
```



```
}

variable "filename11" {
  type = tuple([string, number, bool, list(number)])
  default = ["shaik", 1, true, [1, 2, 3]]
}
variable "filename12" {
  type = object({
    name    = string
    id      = number
    address = list(string)
  })
  default = {
    name    = "default"
    id      = 0
    address = ["A.P", “HYD]
  }
}
```

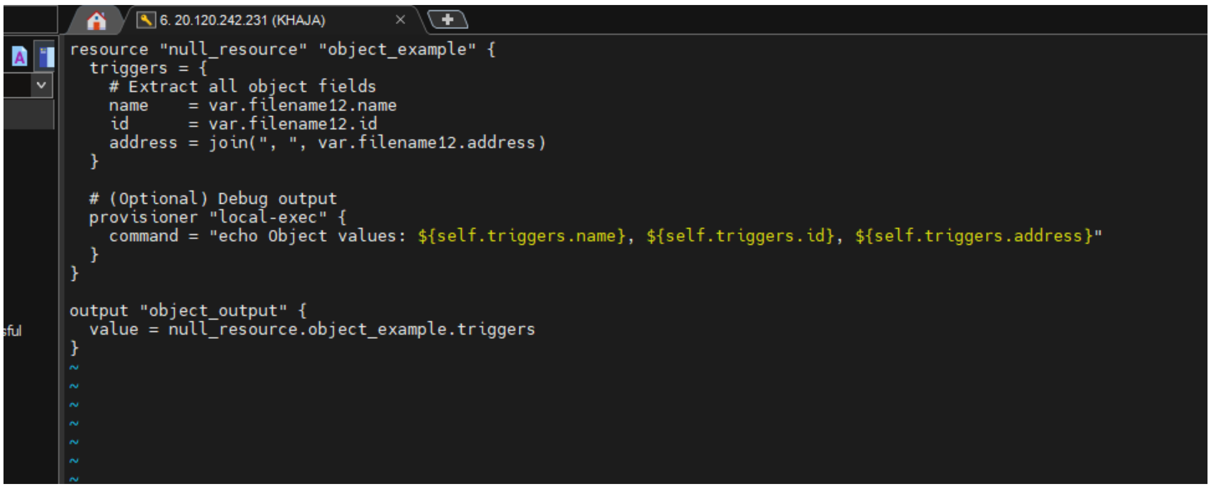
Loading remote monitoring, please wait...

res.tf for declaring reources

```
resource "null_resource" "object_example" {
  triggers = {
    # Extract all object fields
    name     = var.filename12.name
    id       = var.filename12.id
    address = join(", ", var.filename12.address)
  }
```

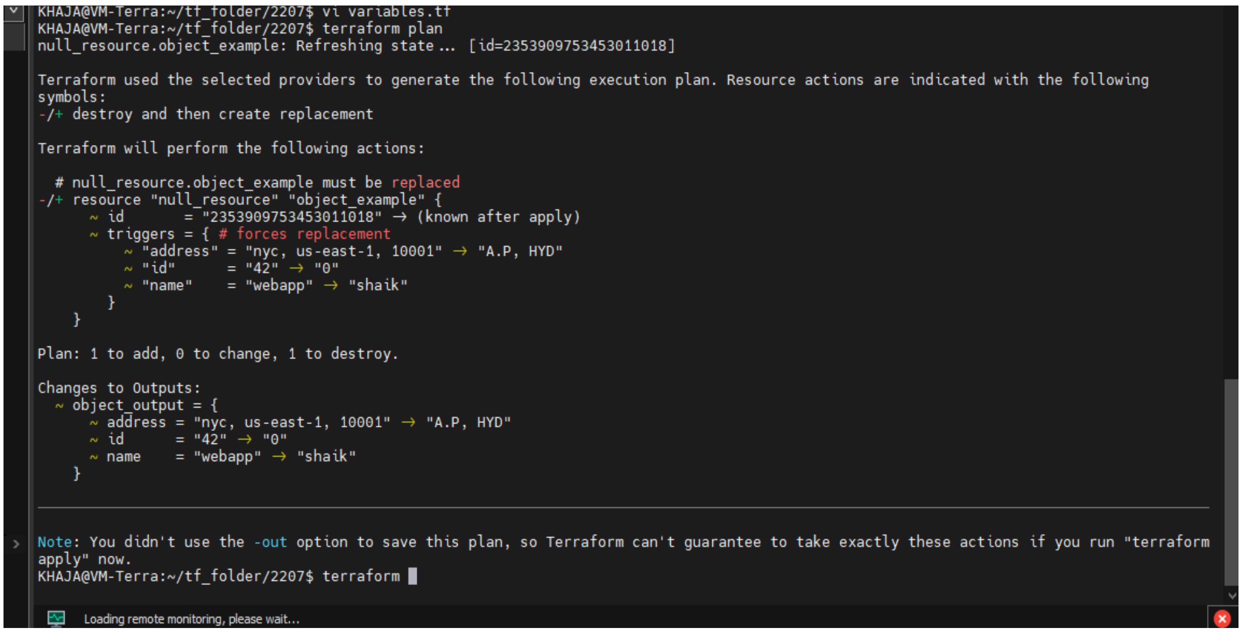
# (Optional) Debug output

```
  provisioner "local-exec" {
    command = "echo Object values: ${self.triggers.name}, ${self.triggers.id},
${self.triggers.address}"
  }
}


output "object_output" {
  value = null_resource.object_example.triggers
}
```



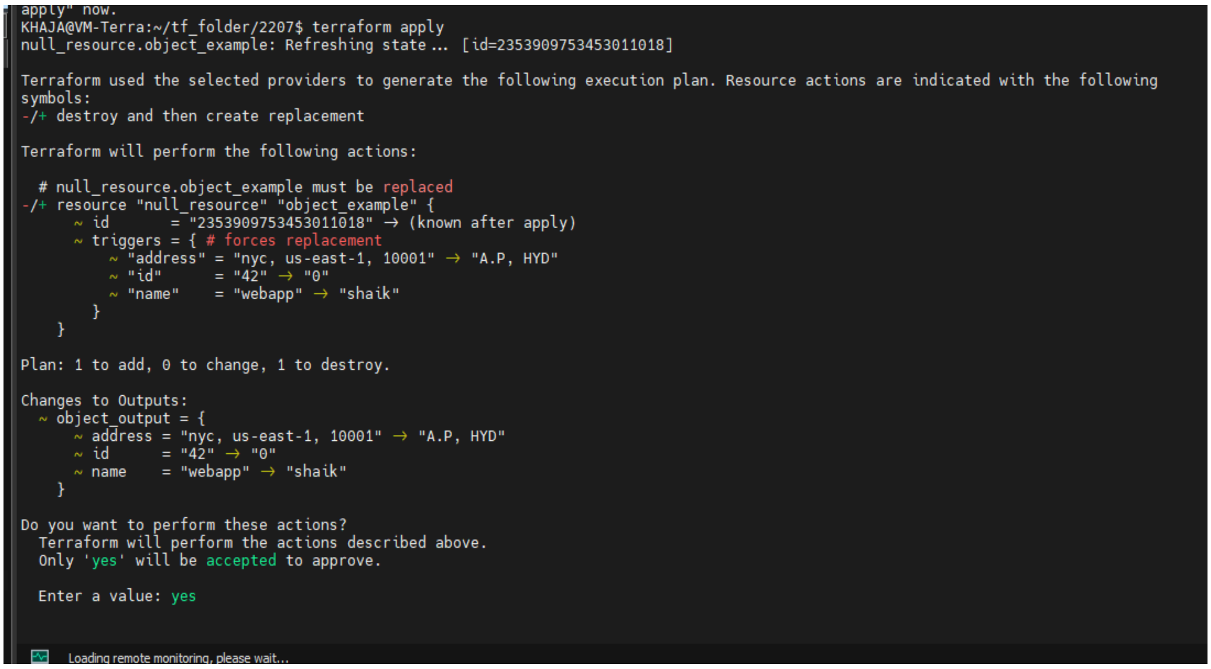
```
resource "null_resource" "object_example" {
  triggers = {
    # Extract all object fields
    name     = var.filename12.name
    id       = var.filename12.id
    address = join(", ", var.filename12.address)
  }

  # (Optional) Debug output
  provisioner "local-exec" {
    command = "echo Object values: ${self.triggers.name}, ${self.triggers.id}, ${self.triggers.address}"
  }
}

output "object_output" {
  value = null_resource.object_example.triggers
}
~
~
~
~
~
~
~
```

Execute the command terraform plan

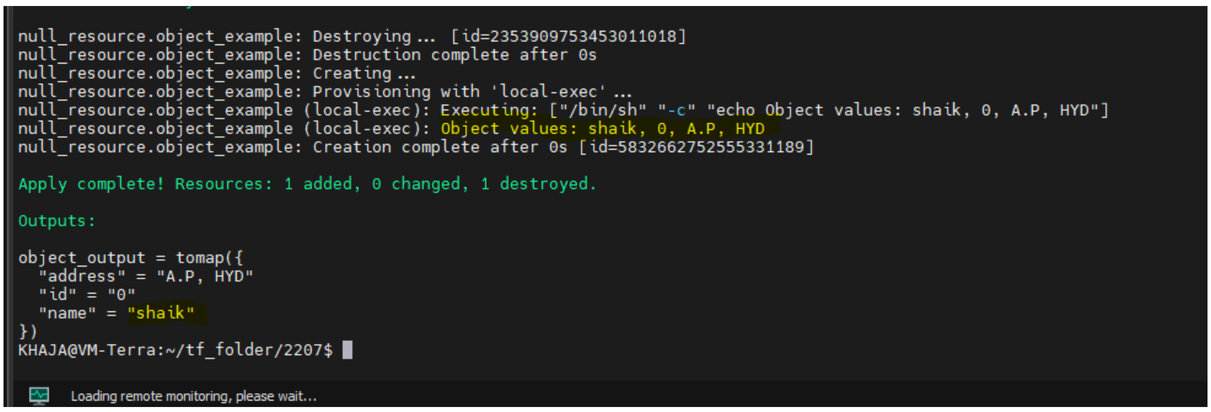


```
KHAJA@VM-Terra:~/tf_folder/2207$ vi variables.tf
KHAJA@VM-Terra:~/tf_folder/2207$ terraform plan
null_resource.object_example: Refreshing state... [id=2353909753453011018]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # null_resource.object_example must be replaced
-/+ resource "null_resource" "object_example" {
      ~ id       = "2353909753453011018" -> (known after apply)
      ~ triggers = { # forces replacement
          ~ "address" = "nyc, us-east-1, 10001" -> "A.P, HYD"
          ~ "id"      = "42" -> "0"
          ~ "name"    = "webapp" -> "shaik"
        }
    }

Plan: 1 to add, 0 to change, 1 to destroy.

Changes to Outputs:
  ~ object_output = {
      ~ address = "nyc, us-east-1, 10001" -> "A.P, HYD"
      ~ id      = "42" -> "0"
      ~ name    = "webapp" -> "shaik"
    }

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform
apply" now.
KHAJA@VM-Terra:~/tf_folder/2207$ terraform
```

Run the command terraform apply for actual change



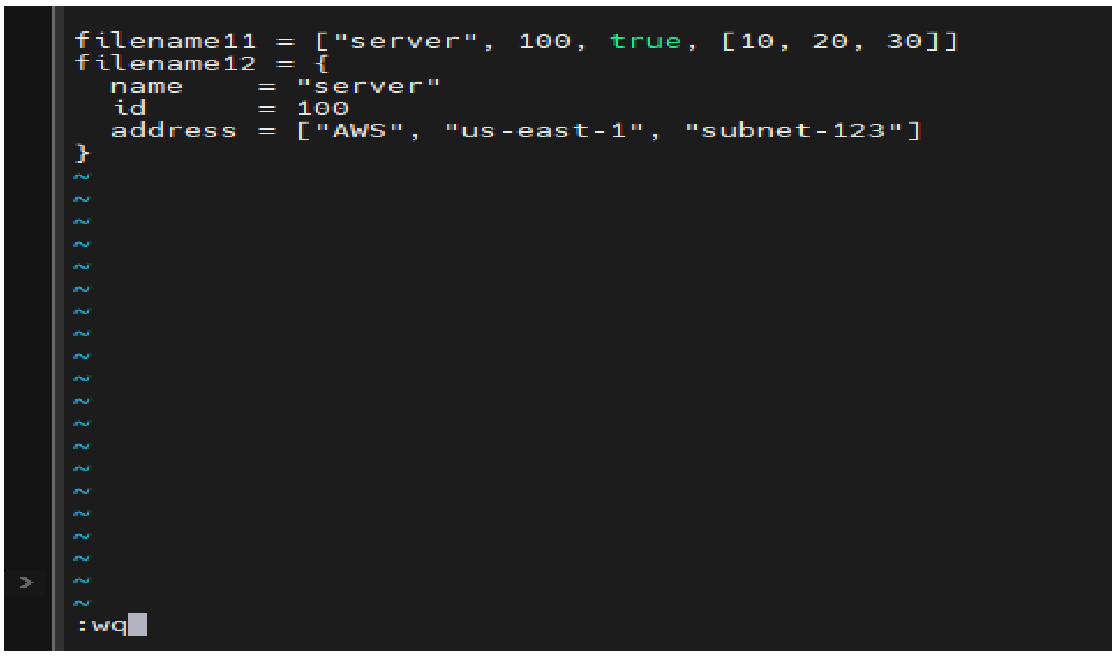
```
apply" now.
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
null_resource.object_example: Refreshing state... [id=2353909753453011018]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # null_resource.object_example must be replaced
-/+ resource "null_resource" "object_example" {
      ~ id       = "2353909753453011018" -> (known after apply)
      ~ triggers = { # forces replacement
          ~ "address" = "nyc, us-east-1, 10001" -> "A.P, HYD"
          ~ "id"      = "42" -> "0"
          ~ "name"    = "webapp" -> "shaik"
        }
    }

Plan: 1 to add, 0 to change, 1 to destroy.

Changes to Outputs:
  ~ object_output = {
      ~ address = "nyc, us-east-1, 10001" -> "A.P, HYD"
      ~ id      = "42" -> "0"
      ~ name    = "webapp" -> "shaik"
    }

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```



```
null_resource.object_example: Destroying ... [id=2353909753453011018]
null_resource.object_example: Destruction complete after 0s
null_resource.object_example: Creating ...
null_resource.object_example: Provisioning with 'local-exec' ...
null_resource.object_example (local-exec): Executing: ["/bin/sh" "-c" "echo Object values: shaik, 0, A.P, HYD"]
null_resource.object_example (local-exec): Object values: shaik, 0, A.P, HYD
null_resource.object_example: Creation complete after 0s [id=5832662752555331189]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

object_output = tomap({
  "address" = "A.P, HYD"
  "id" = "0"
  "name" = "shaik"
})
KHAJA@VM-Terra:~/tf_folder/2207$
```

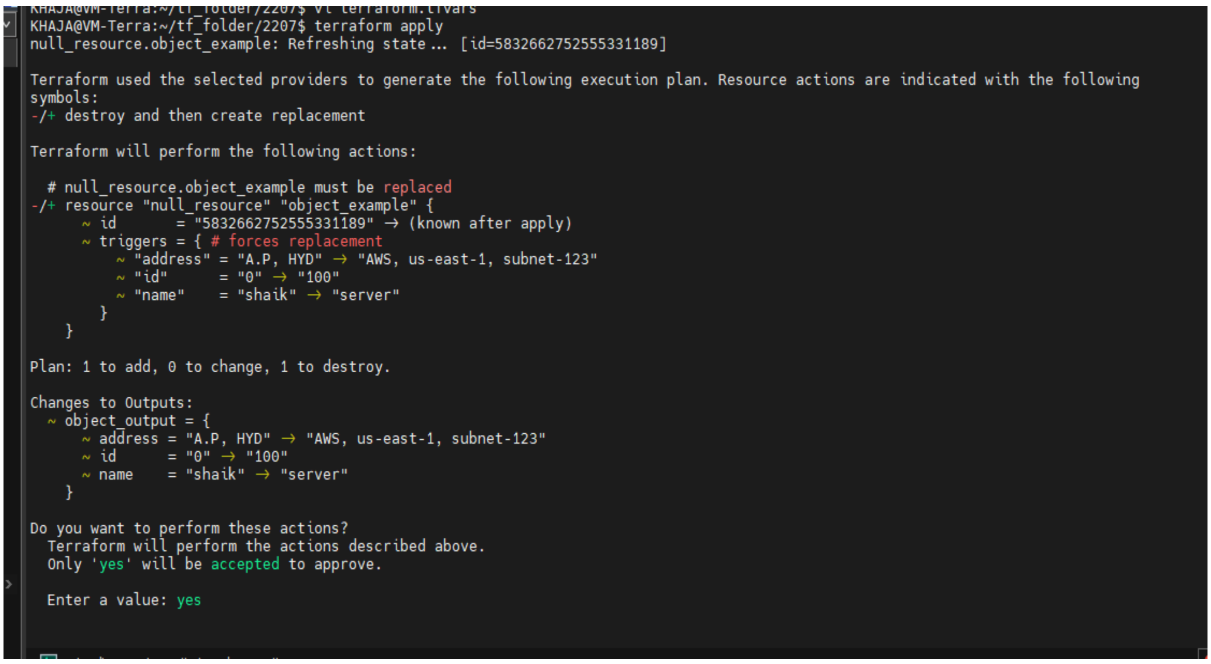
Loading remote monitoring, please wait...

terraform.tfvars (Optional)

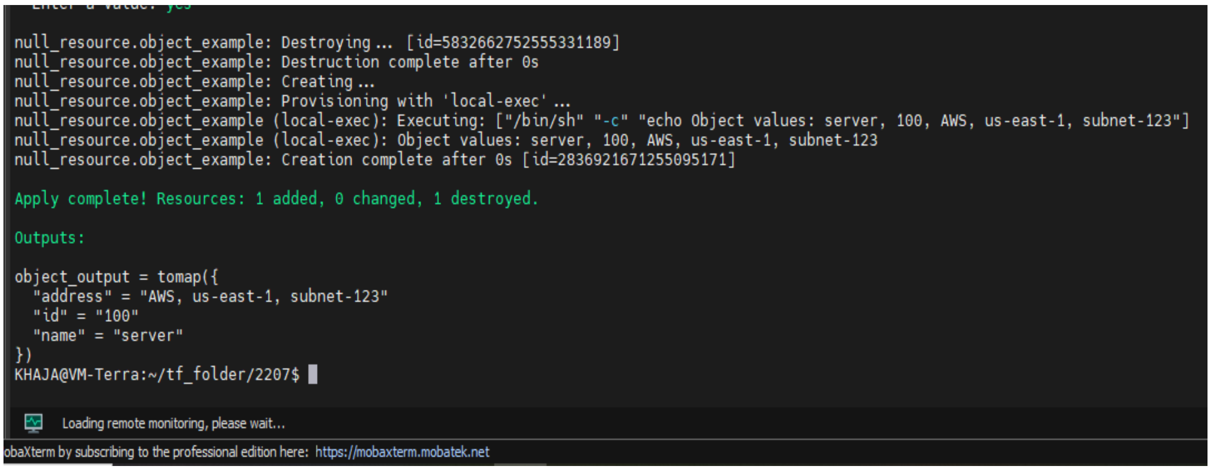


```
filename11 = ["server", 100, true, [10, 20, 30]]
filename12 = {
   name    = "server"
   id      = 100
   address = ["AWS", "us-east-1", "subnet-123"]
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

Run the command terraform apply



```
KHAJA@VM-Terra:~/tf_folder/2207$ vi terraform.tfvars
KHAJA@VM-Terra:~/tf_folder/2207$ terraform apply
null_resource.object_example: Refreshing state ... [id=5832662752555331189]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following
symbols:
-/+ destroy and then create replacement

Terraform will perform the following actions:

  # null_resource.object_example must be replaced
-/+ resource "null_resource" "object_example" {
      ~ id       = "5832662752555331189" -> (known after apply)
      ~ triggers = { # forces replacement
          ~ "address" = "A.P, HYD" -> "AWS, us-east-1, subnet-123"
          ~ "id"      = "0" -> "100"
          ~ "name"    = "shaik" -> "server"
        }
    }

Plan: 1 to add, 0 to change, 1 to destroy.

Changes to Outputs:
  ~ object_output = {
      ~ address = "A.P, HYD" -> "AWS, us-east-1, subnet-123"
      ~ id      = "0" -> "100"
      ~ name    = "shaik" -> "server"
    }

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```



```
null_resource.object_example: Destroying... [id=5832662752555331189]
null_resource.object_example: Destruction complete after 0s
null_resource.object_example: Creating...
null_resource.object_example: Provisioning with 'local-exec'...
null_resource.object_example (local-exec): Executing: ["/bin/sh" "-c" "echo Object values: server, 100, AWS, us-east-1, subnet-123"]
null_resource.object_example (local-exec): Object values: server, 100, AWS, us-east-1, subnet-123
null_resource.object_example: Creation complete after 0s [id=2836921671255095171]

Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

Outputs:

object_output = tomap({
  "address" = "AWS, us-east-1, subnet-123"
  "id" = "100"
  "name" = "server"
})
KHAJA@VM-Terra:~/tf_folder/2207$
```

Loading remote monitoring, please wait...
obaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Key Use Cases for Null Provider with Tuples & Objects

1. **Dependency Management**
   - Ensures resources wait for variable processing before execution.
2. **Debugging Variables**
   - Use local-exec to print variable values during terraform apply.
3. **Triggering Other Actions**
   - Useful with local-exec or remote-exec provisioners when variables change.
4. **Conditional Logic**
   - Combine with count or for_each to control resource creation.